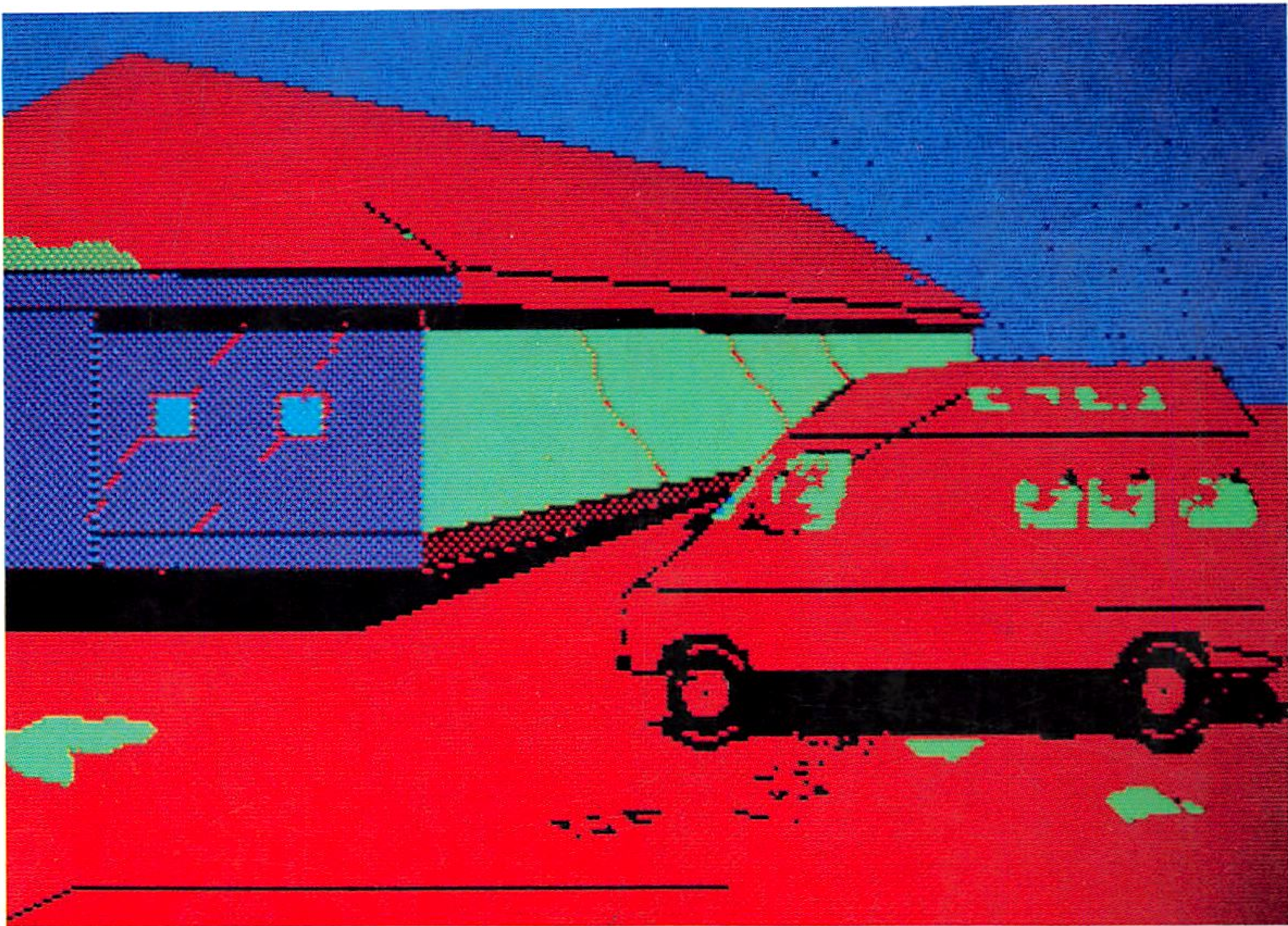
PC-8001mkII

N80-BASIC

REFERENCE MANUAL



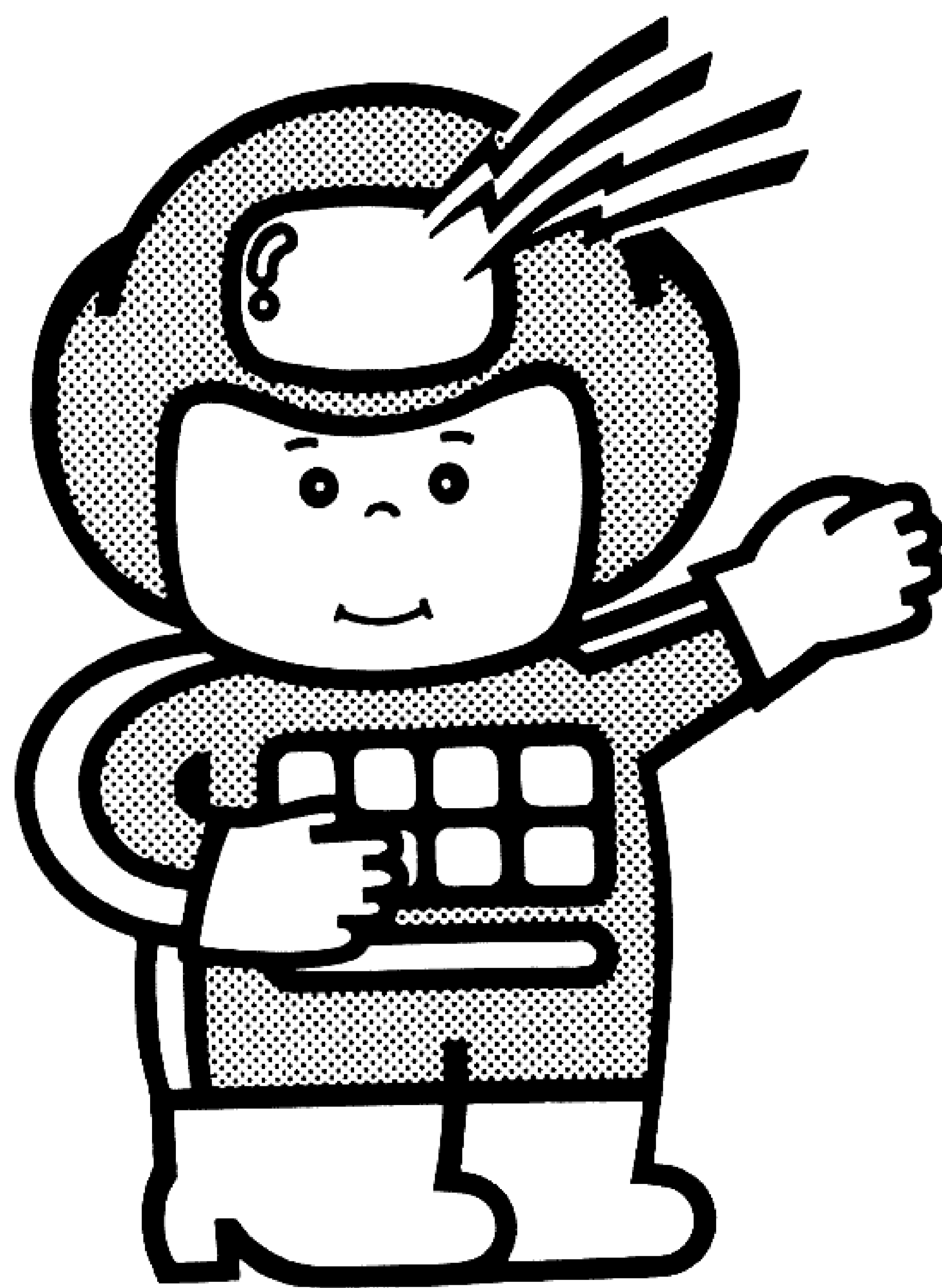
NEC

PC-8001MK2-RM
PTS-121A

PC-8001mkII

N80-BASIC

REFERENCE MANUAL



まえがき

パーソナルコンピュータ PC-8001MKⅡ には，N-BASIC と N80-BASIC という2つの BASIC モードがあります。N-BASIC は，本機の姉妹機種にあたる PC-8001 と同じもので，既に国内では御好評をいただいております。N80-BASIC はその N-BASIC に拡張機能を付加したプログラミング言語で，PC-8001MKⅡ のハードウエアを活かしたプログラムを容易に作成することができます。

本書はこれらの BASIC の命令を個々について，詳細に解説したリファレンスマニュアルです。このマニュアルを読む前に，PC-8001MKⅡ ユーザーズマニュアルをよく読んで，基本的な操作方法を，習得することをお勧めします。

ユーザ各位が，このリファレンスマニュアルを熟読され，優れた機能を持つ BASIC のソフトウエアが，数多く作り出されることを期待いたします。

このマニュアルの構成

本書は各機能，命令などを分かりやすくするため，1，2 章の独立した章と APPENDIX という構成になっています。

また N80-BASIC と N-BASIC の解説は分離せず一括して行います。ただし， N80-BASIC の方が N-BASIC の上位互換性を持っていますから，原則として N80-BASIC を題材にし，特に内容の異なるものや片方にしかない機能の解説の場合はそのつど N80-BASIC，N-BASICと明記してあります。

またサンプルプログラムについては， N-BASIC 専用命令の解説を除いて，すべて N80-BASIC を題材として作られています。実際に試してみる場合は N80-BASIC のモードで実行してください。

1章　N80-BASIC(N-BASIC)の概要

N80-BASIC(N-BASIC) の特徴や機能をコマンド・ステートメントなどの説明に先だって解説しています。その内容は， BASIC 言語の定数，変数の意味，演算の決まりなど， BASIC 言語を使う上での最低限の知識と， N80-BASIC に新しく追加された機能の紹介や，それを使う上での知識です。これらは2章，文法の説明を理解する上でどうしても必要な事項です。必ず読むようにしてください。

2章　コマンド・ステートメント・関数・予約変数

BASIC の機能は大きく分けてコマンド・ステートメントと関数に分類することができます。

コマンド及びステートメントは BASIC が持つ命令で，コマンドは一般にダイレクトモードで使用するもので実行後 “Ok” が表示されます。ステートメントは主にプログラム中で使用します（明確に分かれている訳ではない）。

関数は何らかの命令に付随して引用される機能です。

付　録

ここでは，ファイルの作り方，機械語プログラムの作り方や各種の一覧表などをまとめて取り扱っています。

N$_{80}$-BASIC
リファレンスマニュアル

目　次

第2章　コマンド・ステートメント・関数・予約変数

付　　録

機能別索引

1．一般的な命令

一般コマンド

一般ステートメント

2．入出力に関する命令・関数

入出力コマンド

（フロッピィディスク・カセットテープ）

入出力ステートメント

（フロッピィディスク・カセットテープ）

入出力関数

3．画面制御に関する命令・関数

画面制御ステートメント

画面制御関数

画面制御予約変数

4．グラフィックスに関する命令・関数

グラフィックステートメント

グラフィック関数

5．算術演算に関する関数

数値関数

6．文字列操作に関する命令・関数

7．特殊な命令や関数

特殊コマンド

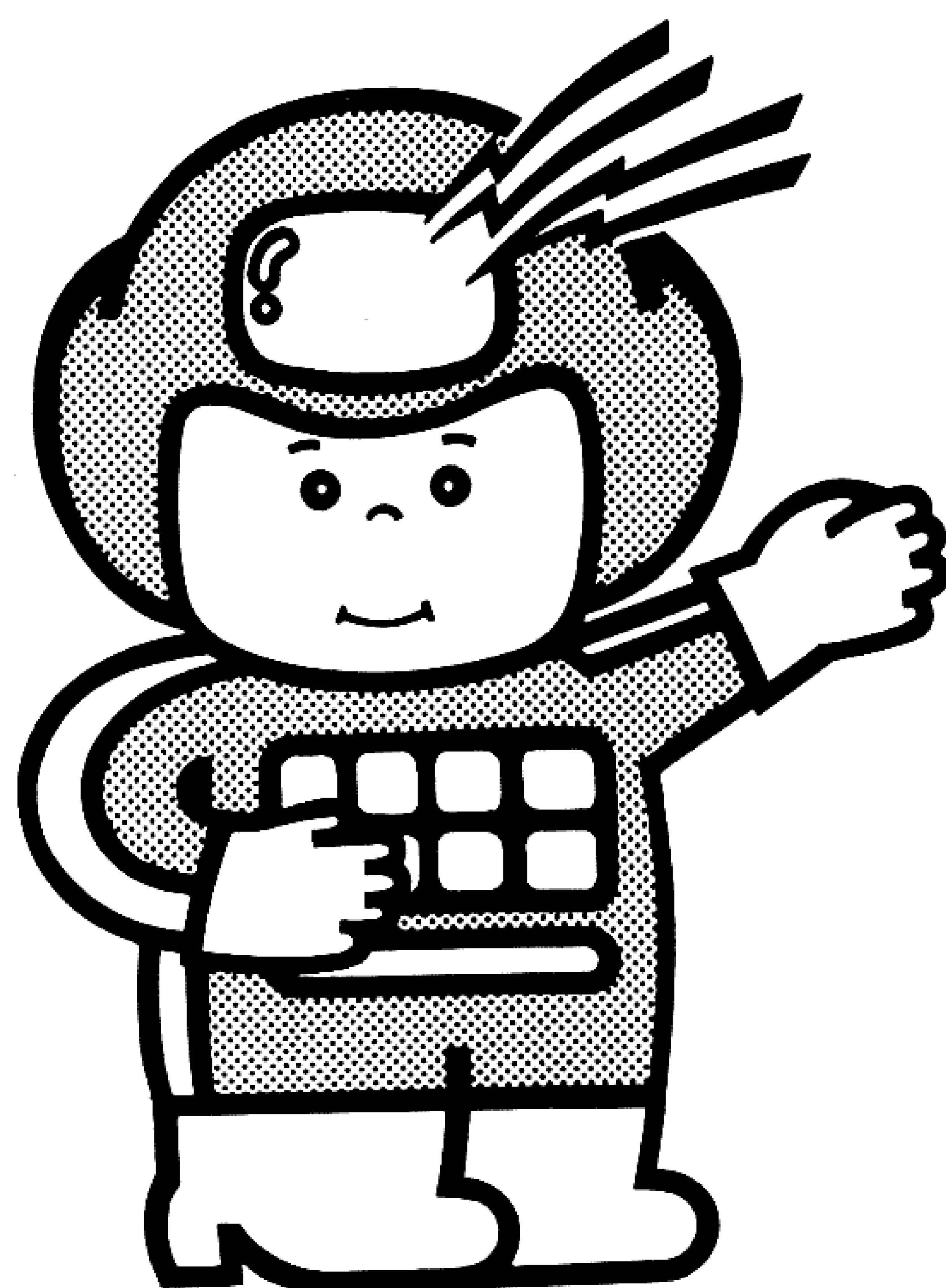
特殊ステートメント

特殊関数

特殊予約変数

第1章

N80-BASIC概要



1.1 N80-BASICとN-BASIC

PC-8001MKII(以下MKIIと呼びます)は，N80-BASICとN-BASICという2つのモードを持っており，必要に応じて切り替えて使用します。

N-BASICは姉妹機のPC-8001に搭載されているものと全く同じで，現在数多く流通しているPC-8001用のソフトウェアを利用して頂くために用意されています。ただし，このモードではMKIIが持っている多くの重宝なハードウェア(高解像度グラフィックスなど)を完全にサポートすることはできません。

N80-BASICは，N-BASICにMKIIが持つハードウェアのすべてを活用するために，御要望の多かった高解像度グラフィックスの機能などいくつかの拡張命令が付加されたものでMKIIのメインプログラミング言語です。

この2つのBASICは完全なコンパチビリティを持っていてN-BASICで動くプログラムならばN80-BASIC上でも変更なしに動きます。ただし，プログラムの容量が大きいとき，または割り込みの優先順位が異なるため各種インタフェースボードを使用した場合などは変更が生じる場合があります。

1.2 N80-BASICの主な特徴

N80-BASICは新しい機能として次のような特徴を持っています。

1. MKIIが持っているハードウェアの機能のほとんどをN80-BASICで使用することができます。

   操作可能な機器を以下に示します。

   ○プログラマブル・ファンクションキー
   ○オーディオカセットテープレコーダ
   ○RS−232Cインタフェース機器
   ○ミニフロッピィディスク
   ○8インチ・フロッピィディスク

○セントロニクス・インタフェース機器(プリンタ)

2. 16Kバイトのグラフィック専用RAMを持っているため，320×200ピクセル(4色カラーモード)，640×200ピクセル(モノクロモード，アトリビュートカラーモード)の高解像度グラフィックスを実現できます。

3. リストなどキャラクタを表示するためのテキストRAMと，グラフィックRAMは完全に区別されているため，テキスト画面とグラフィック画面の合成が簡単にできます。

4. オプションの漢字ROMボードを接続すれば，日本語の文章を処理することができます。

1.3 動作モード

MKⅡの電源を入れBASICを起動すると画面にメッセージが表示されそのあと"Ok"という文字が表示されます。

この"Ok"が表示されている状態を「BASICがコマンドレベルにある。」といいます。BASICがコマンドレベルにあるときは，そのBASICのあらゆるコマンドをキーボードから入力することができます。BASICのコマンドは，ダイレクトモード，プログラムモードのいずれかで使用することができます。

1.3.1 ダイレクトモード

行番号をつけずにBASICの文法に沿った文をキーボードから入力し，RETURN キーを押すとその文はすぐに実行されます。これをダイレクトモードの実行といいます。入力された文はメモリに残りません。

1.3.2 プログラムモード

行番号(0～65529までの整数)をつけて文を入力した場合，RETURN

キーを押しても，すぐには実行されずメモリの中にプログラムとして行番号順に格納されます。この格納されたプログラムは，RUN コマンド，GO TO ステートメント，GOSUB ステートメントにより実行させることができます。これをプログラムモードの実行といいます。なお，1つの行を入力後 **RETURN** キーを押した場合 "Ok" の表示はされませんが，BASIC はコマンドレベルにあります。

1.3.3 ターミナルモード

MKⅡに TERM コマンドを入力すると，他のコンピュータ，あるいは周辺装置の端末として使用することができます。この状態をターミナルモードといいます。

1.4 文

文とは，BASIC が行う手続きを記述している最小単位です。

文には，BASIC が実行する式，ステートメント，コマンド，関数などを書くことができます。また文は，コロン(:)を用いて他の文をつなぐことができます。これは複文(マルチステートメント)と呼ばれ，複文は，1行(行番号も含めて)255文字以内の長さまで許されています。

1.5 行番号

BASIC の各プログラム行は，行番号で始まらなければなりません。行番号には，0〜65529までの整数を用います。行番号はプログラム行をメモリに格納する順序を示し，実行も行番号の若い方から行われます。また分岐や編集の目印としても使用されます。

行番号の代わりにピリオド(.)を使うことができる場合があります。ピリオドは LIST，AUTO，DELETE などコマンド中で，エラー発生，編集

などによってBASIC内のポインタが示している現在の行を表すものです。

(例) **LIST.**

AUTO.

DELETE .-100

1.6　使用できる文字とコントロールキャラクタ

N80(N)-BASICの使用できる文字は，英文字，カナ，数字，特殊記号，そしてグラフィックキャラクタより構成されており，これらはキャラクタセットと呼ばれます。英文字は大文字と小文字で，数字は0から9までです。これらキャラクタセットの詳細は，付録Eの“キャラクタコード”を参照してください。他にも特別の意味をもつコントロールキャラクタがあります。これについては，付録Hの“コントロールコード”を参照してください。

1.7　特殊記号の使い方

BASICでは，演算子(＋，－，＊，／)などの他にも特別な意味を持つ記号があります。ここでその意味をまとめて説明しておきます。

1.　ピリオド(.)

現在BASICが着目している行番号の値を持っておりポインタとして使用することができます。BASICが着目しているとは，新しい行を挿入した，エラーが発生したなどの行です。

(例) LIST.

2.　ハイフン(－)

LIST，DELETEなどで行の範囲を指定する時，何行から何行までという場合に使います。

（例） **DELETE 100-200**

3. コロン(:)

   マルチステートメントの区切りとして使います。

（例） **A=B+C : PRINT A**

4. コンマ(,)

   PRINT，INPUT などパラメータが並ぶ場合その区切りとして使います。

（例） **INPUT A, B, C**

　　　COLOR 7,,,0

5. セミコロン(;)

   PRINT などの区切りとして使います。

（例） **PRINT "A=" ; A**

6. アポストロフィ(')

   REM(リマーク)の代用として使えます。

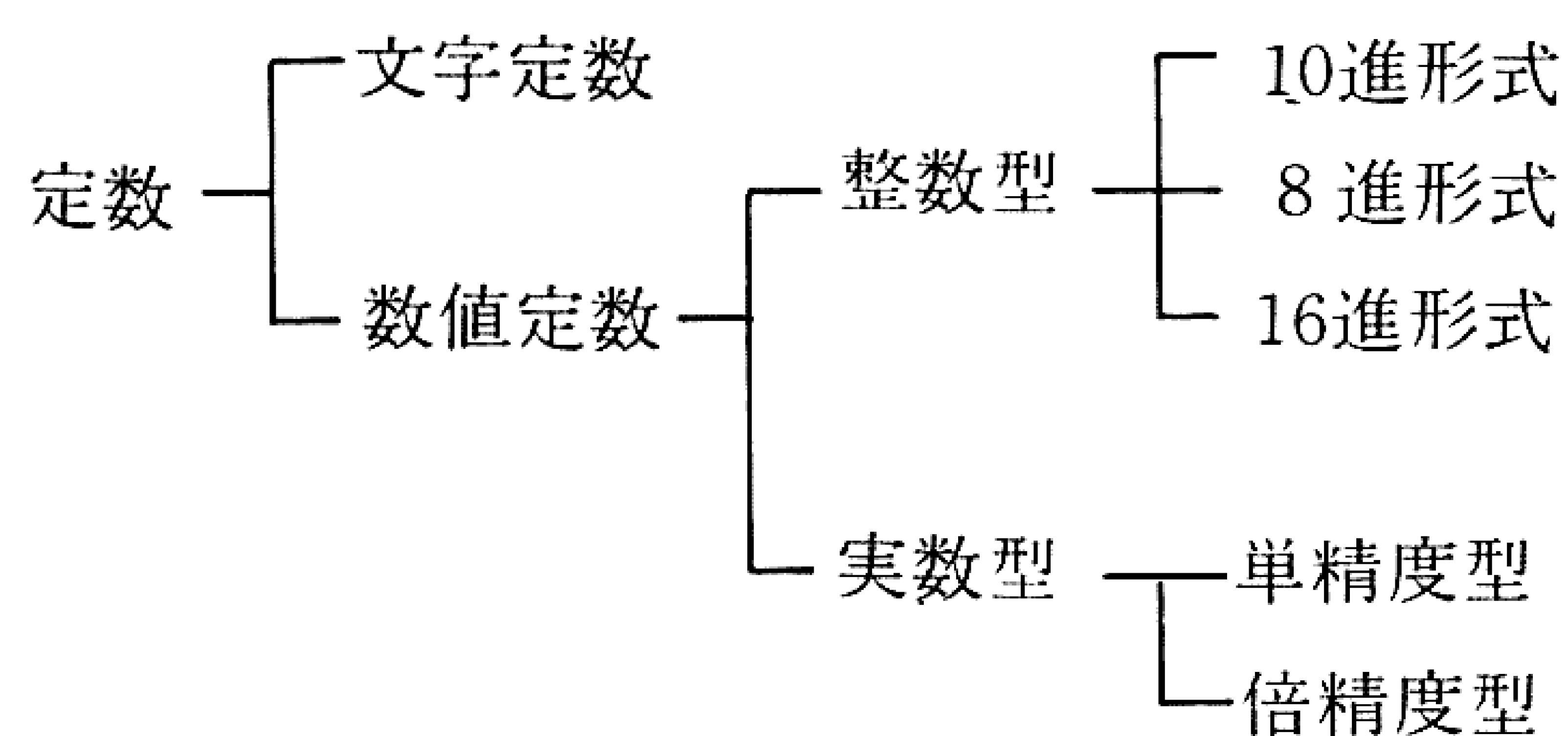
7. クエスチョンマーク(?)

   PRINT の代用として使えます。

1.8 定　　数

1.8.1 定数の種類

N80-BASIC の定数には以下のものがあります。



1.8.2 文字定数

文字定数とは，255文字以下のダブルクォーテーションマーク(”)で囲まれた英数字，カナ文字，記号などの列のことです。また，何もない文字列のことをヌルストリングといいます。なお，(”)を文字列の中に記述する場合は，CHR$関数を用いなければなりません。

例） **“Good Morning”**

“123456789”

“パーソナルコンピュータ”

“”(ヌルストリング)

1.8.3 数値定数

数値定数は，整数型，実数型に分けられ，それらは正あるいは負の数，または0です。

負の数の前には必ずマイナス符号をつけなければなりませんが，正の数の前のプラス符号は省略することができます。

1.8.4 整数型

(1) 10進形式

－32768～＋32767までのすべての整数，または，数の後に％をつけたもの。小数点をつけることはできません。

例） **32767**

－123

32767%←整数を表す。

(2) 8進形式

頭に&Oまたは&を伴った，0から7までの数字の並び。&0～&177777の範囲

例） **&12345**

&07777

(3) 16進形式

頭に &H を伴った，０からFまでの並び。

例） **&H100**

&HFFFF

(**注　意**) 8 進形式または16進形式で入力された数値は，PRINT，LPRINT では10進形式で出力されます。

1.8.5 実数型

実数型は，単精度型と倍精度型に分けられます。

1.8.6 単精度型

有効桁 7 桁の精度で格納されます。出力のときは 7 桁目が四捨五入され，6 桁以下で表示されます。扱える数値は，－1.7014E＋38～1.70141E＋38 です。

(1)　7 桁以下の実数

(2)　E(大文字に限る)を使った指数形式

(3)　最後に！を伴った数

例） **1.23**
−7.09E−06
3525.68
3.14!

1.8.7 倍精度型

有効桁16桁の精度で格納され，16桁以下で表示されます。

(1)　8 桁以上の数

(2)　D(大文字に限る)を使った指数形式

(3)　最後に＃を伴った数

例） **1234567890**

−1.09432D−06＋0.3141592653D＋01

56789.0＃

8657036.1543976

1.9 変　　数

1.9.1 変数とは？

BASIC のプログラム中で使われる値を格納するためのエリアに，英数字から成る名前を対応させたものが変数です。

変数の値は，プログラマによって定義され，演算，参照などに使うことができます。これは演算，代入実行後，割り当てられます。数値変数は，値が割り当てられるまえに参照されたら 0 が代入され，文字変形はヌルストリング(空の文字列)が代入されます。

1.9.2 変数名と型宣言文字

N80(N)-BASICにおいて，変数名は何文字あってもかまいません。ただし N80(N)-BASICでは最初の 2 文字までの判別しかしません。また最初の文字は英字でなければなりませんが，残りの文字は英文字または数字のどれでもかまいません。ただし変数名は付録Jに示す予約語を含んではいけません。

同じ変数名でも型が違えば区別されます。変数の型は型宣言文字によって決まり，型宣言文字は変数名の最後につけて，その変数の型を表わします。型宣言文字を省略すると，“！”がついているとみなされます(単精度実数型変数となる)。

型宣言文字		
	%	整数型
	!	単精度実数型
	#	倍精度実数型
	$	文字型

A
A#
A%
A$
これらは区別されるが，AとA！は同じ。

変数の型を宣言するのにもう 1 つ便利な方法があります。 それは **DEFINT**，**DEFSNG**，**DEFDBL**，**DEFSTR** の型宣言文をプログラム中で用いる方法です。これについては 2 章で詳しく説明します。

1.9.3 配列変数

配列変数は，1つの変数名でいくつかの要素を参照することのできる変数です。配列変数のそれぞれの要素は，整数または整数表記による添字によって参照されます。

配列変数の次元は255次元まで，添字はメモリの範囲内で許されており，これらの大きさはDIM(2章参照)で宣言します。ただし，各添字は原則として0から始まりますから，実際の要素数は添字の数+1となります。

例)	DIM A(10)	1次元配列，要素の数=11
	DIM TA(10, 50)	2次元配列，要素の数 11×51=561
	DIM NA$(2, 5, 3)	3次元配列，要素の数 3×6×4=72

(注　意) 各添字の数が10以下の時はDIM文による宣言を省略することができます。

1.9.4 予約変数

N80(N)−BASICには，BASIC自身が専用に使う予約変数があります。これらの変数は，ユーザによって一般の変数として使うことはできません。

TIME$	現在の時分秒をHH：MM：SSの形でもっています。同じ形で代入することもできます。
DATE$	現在の年月日をYY/MM/DDの形をもっています。同じ形で代入することもできます。
ERL	エラーの生じた時，エラーが発生した行番号をもっています。代入することはできません。
ERR	エラーが発生した時，生じたエラーのエラーコードをもっています。代入することはできません。

1.10 型変換

N80(N)-BASICの数値データは，必要に応じてその形から他の形に変

換することができます。ただし，文字型と数値型の間でこの変換を行うことができません。

(1) ある型の数値データが，違った型の数値変数に代入された場合，数値は，その変数名によって宣言された型に変換されます。

例）
```
10 AB%=1.234
20 PRINT AB%
RUN
 1
```

(2) 精度の違う数値間の演算の場合，精度の高い方に変換されて，演算が行われます。たとえば，10#/3 の場合は，10#/3# として演算が行われます。

例）
```
10 A#=10#/3
20 B#=10#/3#
30 PRINT A#, B#
RUN
3.333333333333333 3.333333333333333
```

(3) 論理演算の場合，扱われる数値はすべて整数に変換され，結果は整数で与えられます。

例）
```
10 A#=12.34
20 B=NOT A#
30 PRINT  B, A#
RUN
-13 12.34000015258789
```

(4) 実数が整数に変換される場合は，小数点以下は切り捨てます。

この時，整数型で扱える範囲を越えた場合はエラーが起こります。

例）
```
10 A%=34.4
20 B%=34.5
30 PRINT A% B%
RUN
34            34
```

```
10 A#=1.234E+07
20 B%=A#
30 PRINT B%, A#
RUN
Overflow in 20
```

(5) 倍精度変数が単精度変数に代入された時は，変数の値は有効数字7桁に丸めたものとなります。単精度変数の精度は7桁であり，もとの倍精度の数値との誤差の絶対値は，6.3E−8以下となります。

例）
```
10 A#=1.23456789#
20 B!=A#
30 PRINT A#, B!
RUN
1.23456789          1.23457
```

1.11 式と演算

式とは定数や変数を演算子で結合した一般的な数式や，単に文字や数値，あるいは変数だけのものをいいます。

例）
```
"BASIC"
3.14
10+3/5
A+B/C-D
TAN(DO)
```

BASICの演算は次の5つに分類されます。

1. 算術演算
2. 関係演算
3. 論理演算
4. 関数
5. 文字列演算

1.11.1 算術演算

算術演算子には次のようなものがあります。

実行順序	演算子	演算	例
↓	^	指数演算	X^Y
	−	負号	−X
	*，／	乗算，実数の除算	X*Y, X／Y
	+，−	加算，減算	X+Y, X−Y

演算の実行順序を変更する場合カッコを使用します。カッコ中の演算子は他の演算子よりも先に実行されます。カッコ内においては通常の実行順序に従っています。

次に実行例を示します。

	代数表記	BASICの表記
1)	$2X+Y$	2*X+Y
2)	$\frac{X}{Y}+2$	X/Y+2
3)	$\frac{X+Y}{2}$	(X+Y)/2
4)	X^2+2X+1	X^2+2*X+1
5)	X^{Y^2}	X^(Y^2)
6)	$(X^Y)^2$	X^Y^2
7)	$Y(-X)$	Y*(-X)

(1) 整数の除算と剰余の計算

整数の除算は¥によって行われます。扱われる数値が実数の場合は，演算が実行される前に小数点以下が切り捨てられます。商も小数点以下が切り捨てられた整数となります。

例）　**10 ¥ 3＝3**　　(**10/3＝3…1**)

　　　23.75¥5＝4　　(**23/5＝4…3**)

剰余の計算はMODによって行われます。扱われる数値が実数の場合は，演算が実行される前に小数点以下が切り捨てられます。結果は整数の割り算の余りです。

例）　**13.3 MOD 4＝1**　　(**13/4＝3…1**)

　　　25.68 MOD 6.99＝5　　(**25/6＝4…1**)

(2) 0での除算

式の実行時に0での除算が行われた場合は，エラーメッセージを出力し，コマンドレベルに戻ります。

また，べき乗の実行時に，0に対して負のべき乗を行った場合も同様になります。

例） PRINT 2/0

Division by zero

PRINT 0^－1

Division by zero

（3） **桁あふれ(オーバーフロー)**

代入や演算の結果がその変数の型内で表現することのできる範囲をこえた場合，桁あふれが発生します。

桁あふれが起こった場合，BASICは "Overflow" エラーを出力します。

例） PRINT 300^300

Overflow

1.11.2 関係演算

関係演算子は2つの数値を比較するときに用います。結果は，真(－1)，偽(0)で得られ，条件判定文などプログラムの流れを変えるのに用いられます(IF 文参照)。

関係演算子	内容	例
＝	等しい	X＝Y
＜＞，＞＜	等しくない	X＜＞Y，X＞＜Y
＜	小さい	X＜Y
＞	大きい	X＞Y
＜＝，＝＜	小さいか等しい	X＜＝Y，X＝＜Y
＞＝，＝＞	大きいか等しい	X＞＝Y，X＝＞Y

（注　意）＝は代入文にも使うので注意すること。

IF 文の中での使い方の例を次に示します。

```
IF X=0 THEN 1000
IF A+B<>0 THEN X=X+1:Y=Y+1
```

1.11.3 論理演算

論理演算子は複数の条件を調べたり，ビット操作やブール演算を行ったりするのに用います。論理演算は，ビットごとに0または1を結果として

与えます。各論理演算の内容を次に示します。

NOT=not（否定）

X		NOT X
1		0
0		1

AND=and（論理積）

X	Y	X AND Y
1	1	1
1	0	0
0	1	0
0	0	0

OR=inclusive or（論理和）

X	Y	X OR Y
1	1	1
1	0	1
0	1	1
0	0	0

XOR=exclusive or（排他的論理和）

X	Y	X XOR Y
1	1	0
1	0	1
0	1	1
0	0	0

INP=implication（包含）

X	Y	X IMP Y
1	1	1
1	0	0
0	1	1
0	0	1

EQV=equivalence（同値）

X	Y	X EQV Y
1	1	1
1	0	0
0	1	0
0	0	1

論理演算も関係演算子のように，プログラムの流れを変えるのに用いられます。この場合，論理演算子は２つ以上の関係演算子と結ぶことができます。

例）（1）IF X<0 OR 99>X THEN 1000
（2）IF 0<X AND X<100 THEN X=0
（3）IF NOT（A=0）THEN 20

(1)Xが負または，99より大きければ，行番号1000へ飛ぶ。
(2)Xが正でかつ，100より小さければ，Ｘに０を代入する。
(3)Aが０でなければ，行番号20へ飛ぶ。

注　意　論理演算子は演算の前に扱う数値を −32768～＋32768までの値をとる2の補数表示の整数に変換します。もし，この範囲外となれば“Overflow”エラーとなります。もし，０(偽)と−１(真)しか与えられなかったなら，論理演算子は０と−１しか結果として与えません。指定された論理演算では，この整数に対しビットごとに演算を行います。したがって，論理演算子はバイトデータをあるビットパターンに照らし合わせて調べることができます。たとえばAND演算子は機器のI/Oポートのステータスバイトの必要なビット以外のすべてのビットをマスクすることに使えます。またOR演算子はある２進数を作るために２つのビットパターンを混在させることができます。

以下に論理演算子がどのように働くかを例をあげて示します。

例）　63 AND 8=8　　$63=(111111)_2$, $8=(001000)_2$
したがって，63 AND 8=$(001000)_2$=8

−1 AND 8=8　　$-1=(1111111111111111)_2$, $8=(001000)_2$
したがって，−1 AND 8=8

12 OR 11=15　　$12=(1100)_2$, $11=(1011)_2$

したがって，12 OR 11=$(1111)_2$=15

32767 OR−32768=−1

32767=$(0111111111111111)_2$

−32768=−$(1000000000000000)_2$

したがって，32767 OR−32768=

$(1111111111111111)_2$=−1

12 XOR 11=7　　$12=(1100)_2$,　$11=(1011)_2$

したがって，12 XOR 11=$(0111)_2$=7

10 XOR 10=0　　$10=(1010)_2$

したがって，10 XOR 10=$(0000)_2$=0

NOT X=−(X+1)　任意の数の補数表示は1の補数に1を加えたものである。

1.11.4 関　数

関数とは与えられたある引き数に対して，ある決った演算を行うもので，値としては，この演算の結果を返します。

BASICは"組み込み関数"としてSIN（正弦），SQR（平方根）などの数値関数やCHR$，LEFT$などの文字関数を本体内にもっています。

また，BASICは"ユーザ定義関数"としてユーザが自由に定義できる関数機能をもっています。これは2章の"DEFFN"の項で説明します。

一般に引き数に整数しかとらないものは，小数部分を切り捨ててから演算を行います。

次に関数の使い方の例を示します。

例）
```
A=SIN(3.14)+COS(3.14)
PRINT2, 2*2, SQR(2)
```

1.12 文字列の演算

1.12.1 文字列の連結

文字列は演算子＋によって連結することができます。

例）
```
10 A$="NEC" : B$="PC-" : C$="8001MK2"
20 D$=A$+" "+B$+C$
30 PRINT D$
RUN
NEC PC-8001MK2
```

1.12.2 文字列の比較

文字も数値の比較に用いられるものと全く同じ関係演算子を用いて比較することができます。

＝，＜，＞，＜＞，＞＜，＜＝，＝＜，＞＝，＝＞

文字列の場合それぞれの文字列の最初から1文字ずつ，文字の比較を行います。もし相互に全く同じ文字列の場合は，その2つの文字列は等しくなりますが，1箇所でも違った場合は，その文字のキャラクタコード(付録のキャラクタ参照)の大きい方の文字列が大きくなります。文字列の片方が短かくて比較が途中で終った場合は，短かい文字列の方が小さくなります。

文字列の比較においては空白なども意味をもちますから注意してください。

例）
```
"AA"<"AB"
"BASIC"="BASIC"
"X&">"X#"
"PEN△">"PEN"
"cm">"CM"
"DESK"<"DESKS"
```

このように，文字列の比較は文字列の内容を調べたり，文字をアルファベット順に並べたり(ソート)することに使うことができます。

1.13 演算の優先順位

演算は次の順位によって行われます。

1. カッコで囲まれたもの
2. 関数
3. 指数(べき乗) ^
4. 負号(−)
5. *, /
6. ¥
7. MOD
8. +, −
9. 関係演算子(<, >, =など)
10. NOT
11. AND
12. OR
13. XOR
14. IMP
15. EQV

1.14 エラーメッセージ

N80(N)-BASICは，プログラムの実行を中断させなければならないようなエラーを実行時に検出した時，エラーメッセージを表示してコマンドレベルに戻ります。

ダイレクトモードでのエラーメッセージの書式は，

XX

プログラムモードでの書式は，

XX in NNNNN

XX はエラーメッセージで，NNNNN はエラーが検出された行番号です。

N$_{80}$(N)-BASICのエラーコードとエラーメッセージの内容については，付録の"エラーコード表"を参照してください。

1.15 画面モード

ここでいう画面モードとは，文字を表示するテキスト画面の表示桁数，グラフィックスの解像度，白黒モードかカラーモードかなど画面に対する操作のすべての意味を含むものです。N$_{80}$(N)-BASICは，この画面モードを必要に応じて選ぶことができます。

ただし，N$_{80}$-BASICはN-BASICのすべての画面モードに加えて，高解像度グラフィックスモードを持っています。

1.15.1 N-BASICの画面モードと低解像度グラフィックス

(1) 文字表示桁数とグラフィックスの解像度

N-BASIC で許される1画面あたりの表示行数は，20行と25行です。また1行に表示できる桁数は，80，72，40，36，桁が許されています。

N-BASIC のグラフィックスは，キャラクタを横2ドット，縦4ドットに分割したセミグラフィックです。N$_{80}$-BASIC のもつ高解像度グラフィックスに対して，このグラフィックスを低解像度グラフィックスといいます。低解像度グラフィックスでは根本的に文字を表示するテキスト画面とグラフィックスの画面を区別していません。従って低解像度グラフィックスの解像度は，必然的にキャラクタの表示桁数と行数に依存します。例えば80×25文字の画面モードの場合，(80×2)×(25×4)⇨160×100の解像度となります。その他の表示桁数の場合も同様に計算できます。次の表は表示文字数とグラフィックスの解像度の関係です。これらの設定はWIDTHを使って行います。

テキスト画面と低解像度グラフィック画面の解像度

表示文字数(横×縦)	解像度(横×縦)
36×20	72×80
40×20	80×80
72×20	144×80
80×20	160×80
36×25	72×100
40×25	80×100
72×25	144×100
80×25	160×100

(2) 白黒モードとカラーモード

N-BASICはどの文字表示桁数の時でも，白黒モードとカラーモードを選ぶことができます。

カラーモードの場合，8色のカラーを文字，ピクセルに使うことが可能です。ただし，前記したようにN-BASICの低解像度グラフィックスは，キャラクタに依存するセミグラフィックス方式ですから，1キャラクタ内の複数のピクセルを違う色で指定することはできません。また1行中で，20回以上違う色を指定しても無視されます。

白黒モードの場合は，ブリンクやリバースなど6つのファンクション(機能)をキャラクタ単位に設定することができます。

これらの実際の設定の仕方は2章のCOLOR，CONSOLEを参照してください。

1.15.2 N80-BASICの画面モードと高解像度グラフイックス

(1) テキスト画面とグラフィックス画面

N80-BASICでは，リストなどキャラクタ文字を表示するテキスト

画面と高解像度グラフィックスを表示するグラフィックス画面を完全に区別しています。従って個別にそれらのモードを設定する命令があり，片方のモードを変化させても，もう片方が影響を受けることはありません。

(2) テキスト画面

このテキスト画面の機能及びその設定はN-BASICのテキスト画面の場合と同様ですから，1.15.1の“N-BASIC の画面モードと低解像度グラフィックス”を参照してください。

(3) 高解像度グラフィックス画面

N_{80}-BASIC のグラフィックス画面は大量のメモリ(16K)によって構成されており4つのモードを選ぶことができます。ここでそれぞれのモードについて説明します。

• モノクロード

640×200ピクセルの解像度で使える色は黒と，8色のうちいずれか1色の合計2色です。

• アトリビュートカラーモード

640×200ピクセルの解像度でテキスト画面の色と8色のうちいずれか1色が使用できます。

• 4色カラーモード 0

320×200ピクセルの解像度で1つのピクセルに4色の色が指定できます。その色は3色(黒，赤，緑)が固定され残りの1色は8色の中から任意の1色が選択できます。

• 4色カラーモード 1

4色カラーモード0と同様4色の色が指定できます。ただしこのときの色は青，マゼンタ，シアンの3色が固定され，残りの1色は8色の中から任意の1色が選択できます。

これらの画面モード(グラフィックモード)はCMD SCREEN で指定されます。

1.16 グラフィック座標系

1.15の画面モードの所で説明したように，低解像度グラフィックスと高解像度グラフィックスでは解像度が異なっています。従っておのずとそれぞれの座標系も異なってきます。

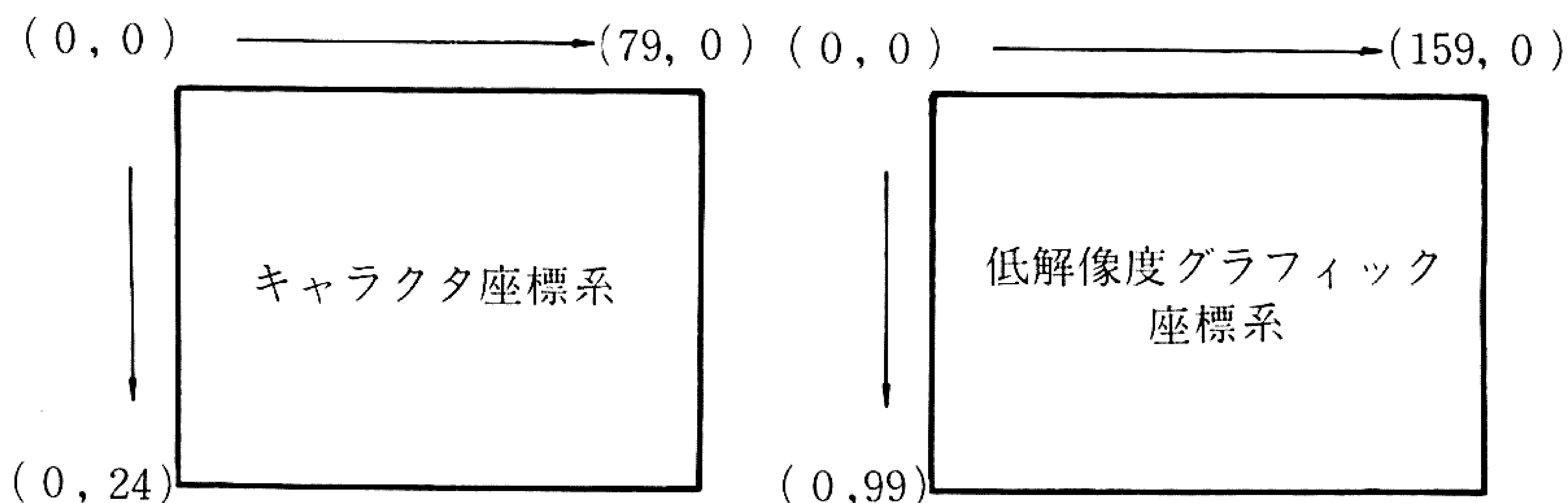
1.16.1 低解像度グラフィック座標系

低解像度グラフィックスはN－BASIC，N_{80}－BASICのどちらのモードでも使用できます。低解像度グラフィックスは，前記したようにキャラクタに依存するセミグラフィック方式ですから，画面上の表示文字数によりピクセルの解像度が決まります。横の文字数をx とし，縦の行数を y とすると画面の解像度は次式で求まります。

横の解像度 (Lxn)＝2＊x

縦の解像度 (Lyn)＝4＊y

次に各座標の画面に対する位置ですが，キャラクタ座標，低解像度グラフィック座標とも画面の左上が原点で(0， 0)です。80×25の文字数を設定したとすると各座標の位置は次の図のようになります。座標の最大値は，原点が(0， 0)から始まることにより，必ず各解像度－１という値になります。



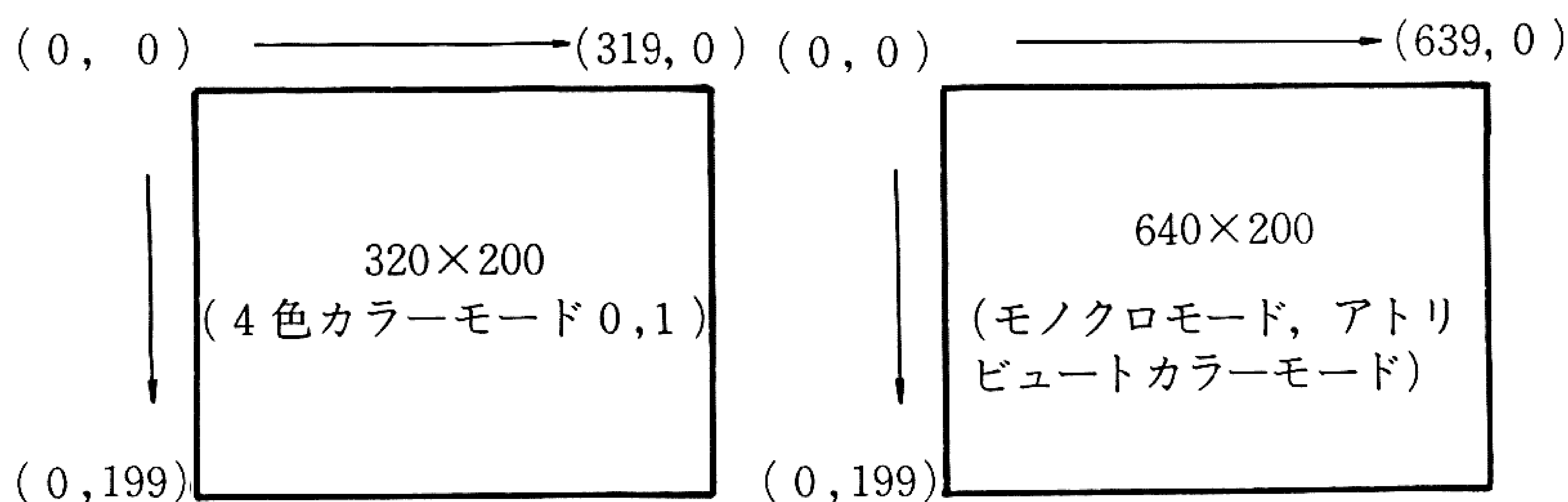
80×25設定時の各座標系

1.16.2 高解像度グラフィック座標系

高解像度グラフィック座標系も，画面の左上が原点(0， 0)であり，基

本的な考え方は低解像度グラフィックスと同様です。ただし，テキスト画面とは分離されていて，相互干渉することはありません。

高解像度グラフィックスの解像度は，320×200と640×200の２つですから各モードの座標系は次の図のようになります。



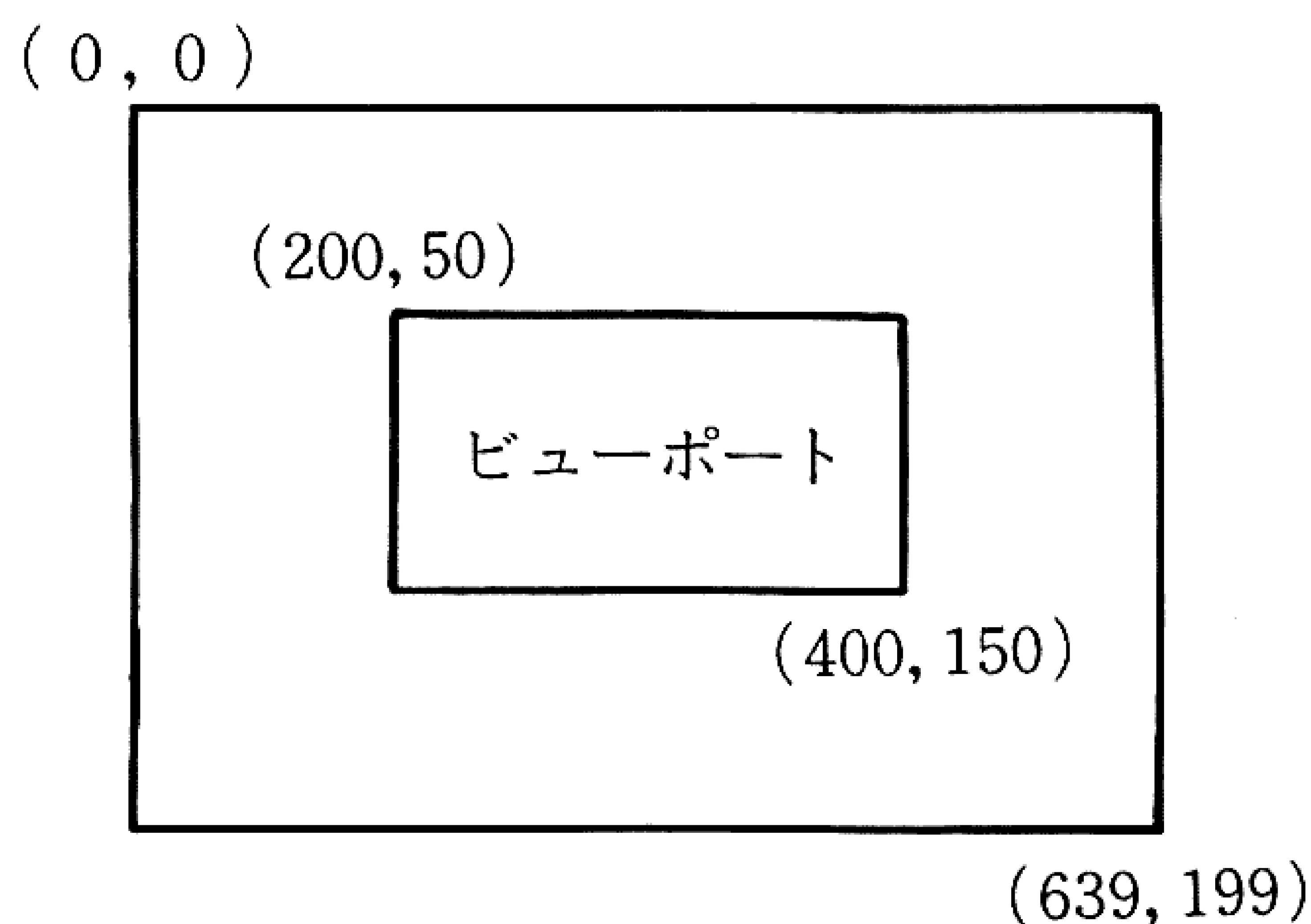
高解像度グラフィック座標系

1.17 ビューポート

1.17.1 ビューポート

N80-BASICでは高解像度グラフィック画面内に“ビューポート”を設定することができます。

ビューポートは，ディスプレイ画面上のどの領域で設定された領域を表示するかの設定です。この操作は，後述するCMD VIEWによって行われます。

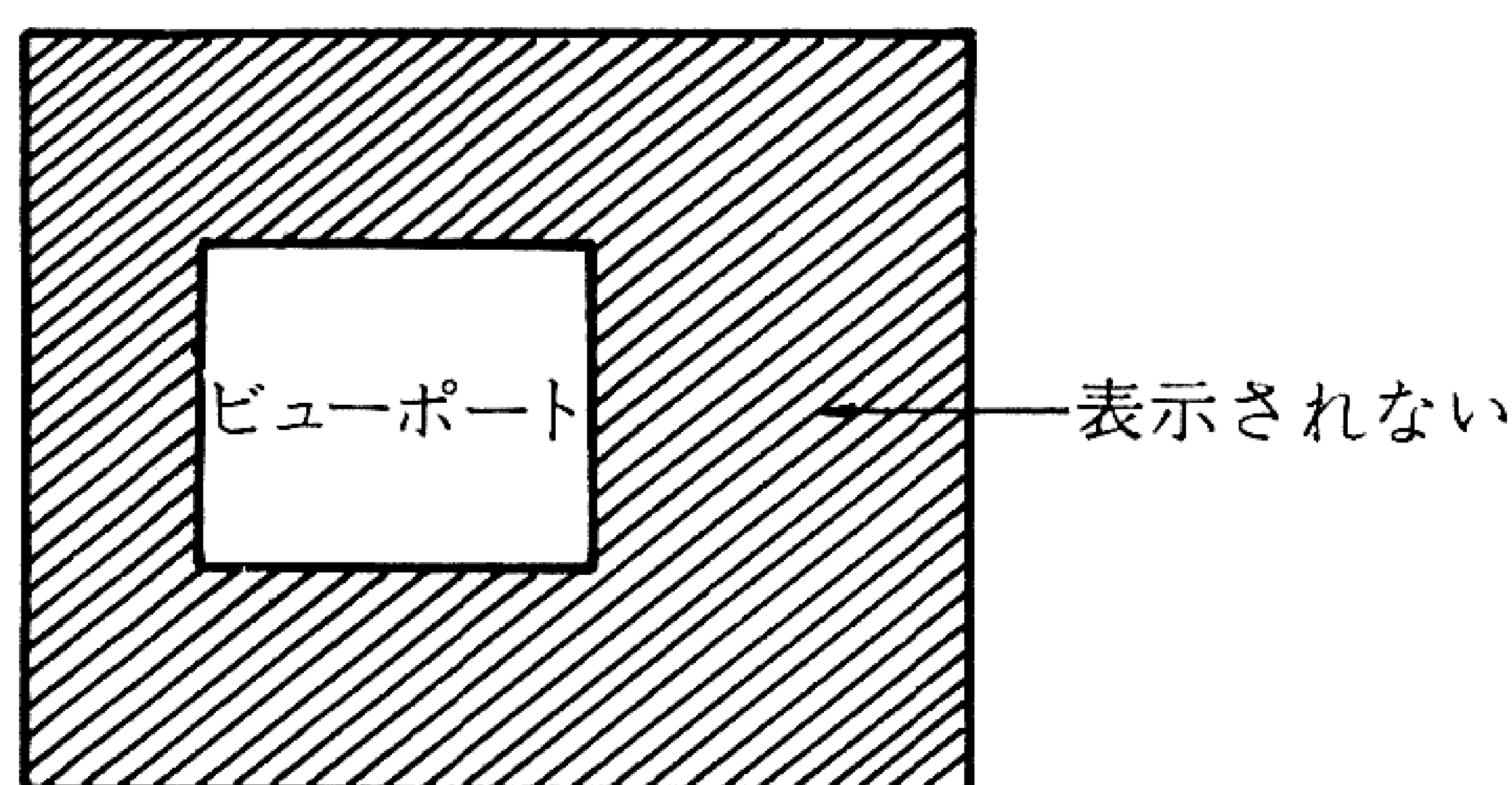


この図では，絶対座標系の(200, 50)から(400, 150)という対角線をもつ四角形をビューポートに設定しています。

このようにビューポートとはディスプレイ画面上でグラフィックスを表示することのできる領域なのです。一度ビューポートを設定すると，その外側のグラフィック画面には，いっさい表示されることはありません。(但し，以前に表示されていたものはそのまま残っています。)

従ってビューポートを設定するということはモニタのディスプレイ画面の大きさを設定するのと同じと考えることもできます。

ディスプレイ画面

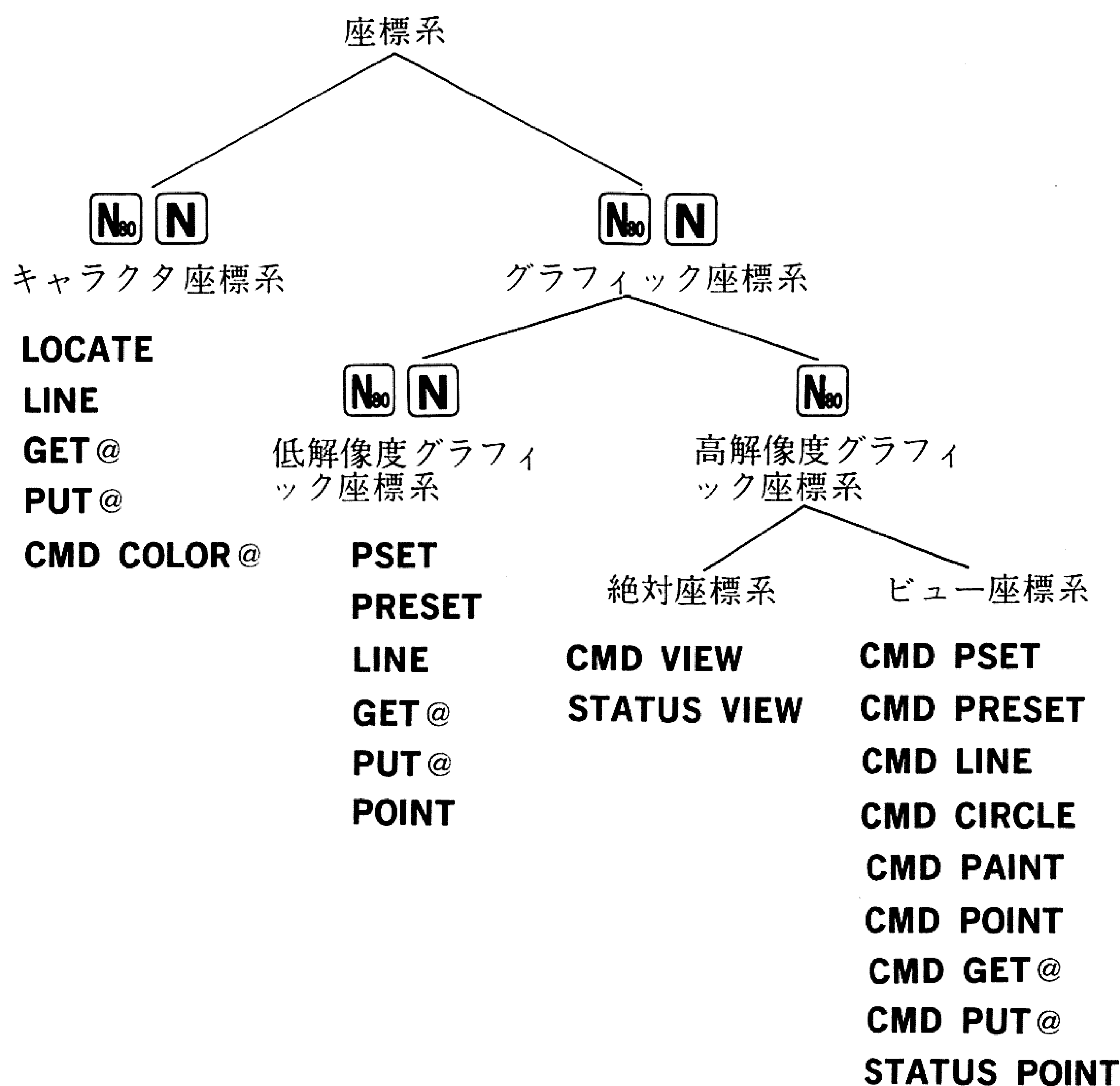


このビューポート内の座標系をビュー座標系といいます。このビュー座標系はビューポートの中にのみ存在するもので，この座標系の各点はディスプレイ画面の１ピクセルに対応しています。この座標系では，ビューポートの左上の頂点が必ず原点(０, ０)に設定され，そこから画面上の各ピクセルを座標の１点として物理的に展開されます。

またビューポートという概念は N_{80}-BASICでのみ通用するものです。従ってN-BASICには，ビュー座標系というものは存在しません。

1.17.2 座標系のまとめ

ここで色々な座標系の整理をしておきましょう。次に示す図はそれぞれの座標系の関係です。また各座標系の下に列挙されているのは，第２章で説明する命令のうちで，その座標系に対して働くものです。



キャラクタ座標系

テキスト画面に存在する座標，座標の1点は1キャラクタに対応する。

グラフィック座標系

グラフィック画面に存在する座標，以下のすべての座標を含んだ意味で使われる概念的なもの。N80(N)-BASICの場合は，各ピクセルに対応した座標として使われる。

ビュー座標系

ビューポートを設定した時に，ビューポート内に存在する座標。

絶対座標系

ディスプレイ画面のすべてをビューポートと考えた時のビュー座標系。

1.18 座標の指定の仕方

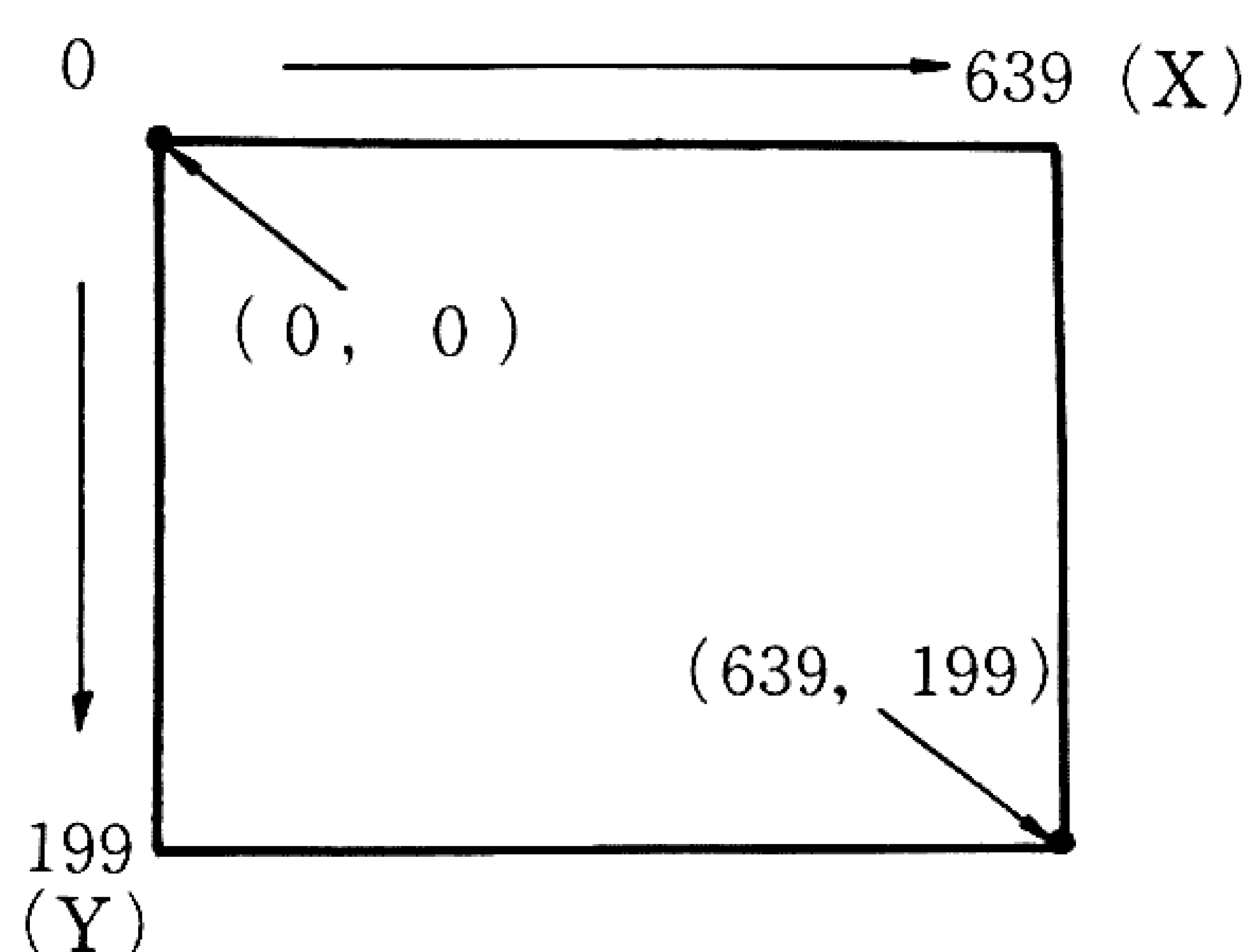
1.17で色々な座標系を説明しましたが，ここではそれらの座標系に対して種々のグラフィック命令を使う場合の“座標の指定の仕方”を説明します。

最も一般的な座標の指定は“絶対座標”によるもので，N₈₀(N)-BASICでの座標指定は，普通どの座標系(キャラクタ座標系，ビュー座標系)に対してもこの指定により行います。ただし，N₈₀-BASICのビュー座標系に対しての座標の指定には，相対座標による方法も許されています。

次にこの２つの指定の仕方について説明します。

1.18.1 絶対座標による指定の仕方

絶対座標とは，X座標，Y座標が決まることによって一意的に決まる座標のことです。例えば，下図のような座標があったとすると，左上の頂点は常に(０，０)であり，右下の頂点は常に(639，199)です。“絶対座標による指定”とはこの絶対座標を使って位置を表す方法をいいます。



絶対座標

これは最も一般的な指定法で，N₈₀(N)-BASICが持つすべての座標系に対して有効です。２章で解説するグラフィック命令などにおいて座標指定する場合，普通はこの方法により行います。例えば，CMD PSET(Hx, Hy)という命令はビュー座標系内の１点にピクセルをセットする命令ですが，

これは次のように書くことができます。

CMD PSET(100, 100)

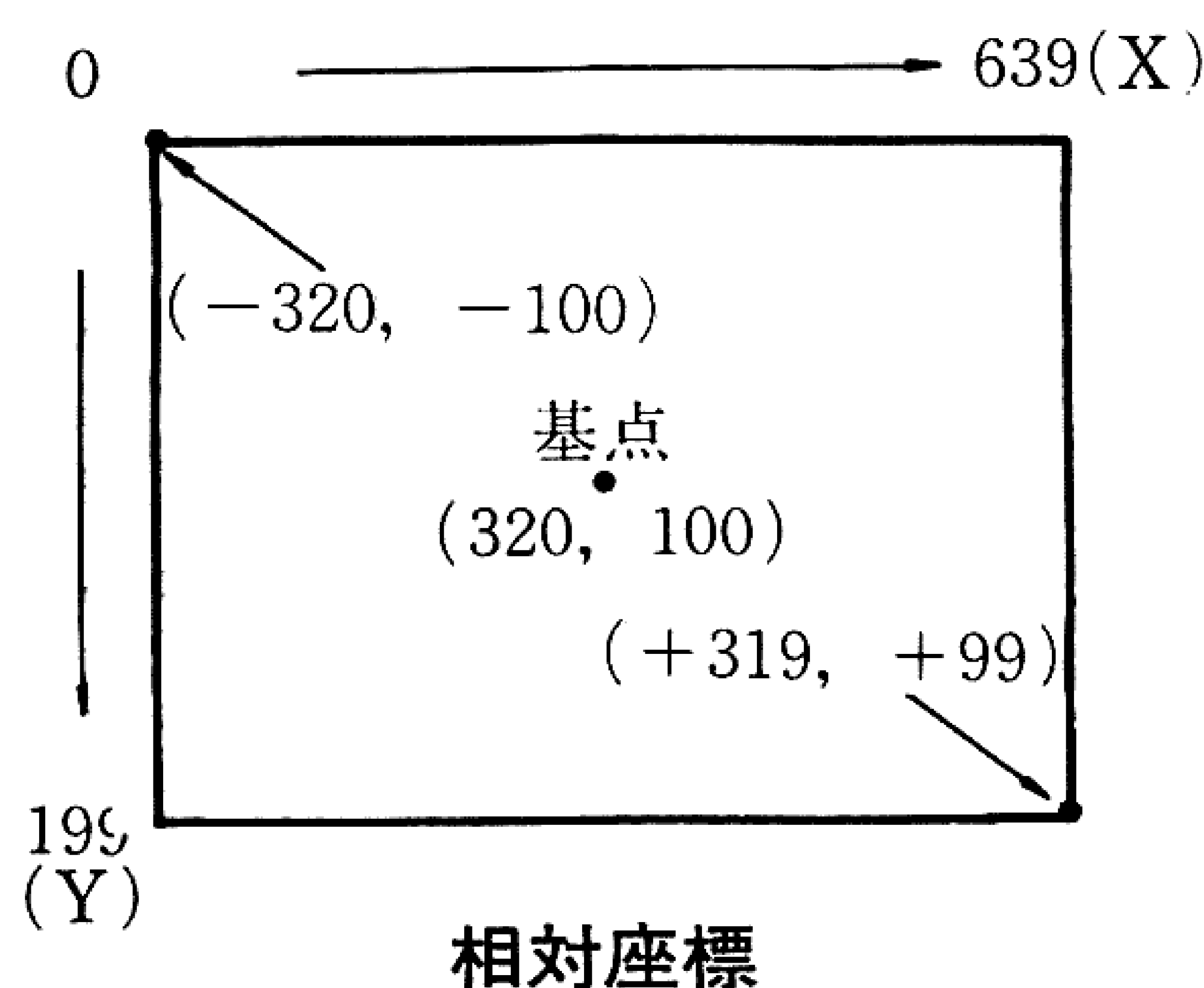
この場合，ビュー座標の(100, 100)にピクセルがセットされます。その他の命令の場合も絶対座標による指定では，すべてこのように(X, Y)という形になって表されます。

1.18.2 相対座標による指定の仕方(N80-BASIC)

N80-BASICの座標系に対するグラフィック命令での座標指定には，“相対座標による指定”が許されています。

“相対座標”とは，座標系内のある1点を基点とした時，X方向，Y方向にどれだけ相対的に移動するかによって決められる座標です。“相対座標による指定”とは，この相対座標を使って位置を表す方法をいいます。

例えば，次のような座標系があり，基点が(320, 100)に設定されていたとすると，左上の頂点は(−320, −100)であり右下の頂点は(+319, +99)となる訳です。



相対座標

この指定法の優れている点は，相対移動量を表す(x, y)は常に同じ値であっても，基点の位置を動かすことによって違った位置を表現することができることです。

N80-BASICはこの基点に当たるものとして，Last referenced Point (以降“LP”と略す)を持っています。このLPは最後に参照された座標という意味で，グラフィック命令で座標を与えた場合，最後に指定された座標を値

として持ちます。

例えば，CMD PSET(Hx, Hy) という命令が実行された場合，LP は (Hx, Hy) という座標に設定されることになります。(2 章，CMD POINT, STATUSPOINT参照。)

N₈₀-BASIC のグラフィック命令で相対座標を使って座標指定を行う場合は，次のような書式になります。

例）　CMD PSET STEP(x, y)

これはドットをセットする命令ですがその他の場合も同様で，座標を表す (x, y) の前に "STEP" を付けて (x, y) が相対量であることを表します。

N₈₀-BASIC のグラフィック命令で相対座標の指定が許されているのは次に示す命令です(おのおのの命令の機能については 2 章で説明します)。

CMD PSET	**CMD PRESET**
CMD LINE	**CMD PAINT**
CMD CIRCLE	**CMD POINT**
CMD GET@	

1.19　カラー

1.19.1　N-BASICのカラー

N-BASICのカラーは黒，青，赤，マゼンタ，緑，シアン，黄，白の 8 色を使うことができ，それぞれ下記のカラーコードによって指定します。

0	1	2	3	4	5	6	7
黒	青	赤	マゼンタ	緑	シアン	黄色	白

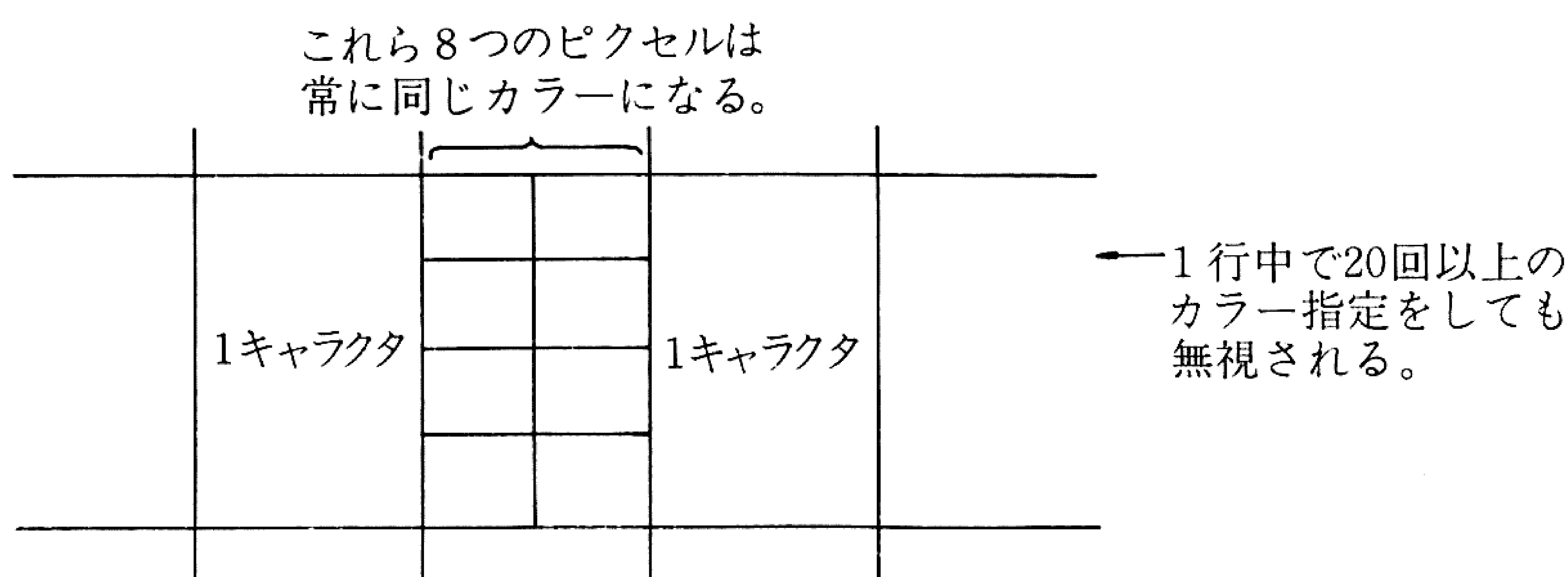
カラーコード

ただし，これらは，"画面モード"の所で説明したように，あくまでキャラクタ単位でしか指定することはできません。従って，キャラクタ文字を 1 文字ずつ塗り分けるということはできますが，低解像度グラフィックス

の場合1キャラクタ内の隣接する複数のピクセルを違う色で塗り分けることはできません。

その他，1行中で20回以上のカラー指定を行った場合，それ以降のカラーは無視され，最後に指定したカラーが有効となります。

次の図は以上の説明を示したものです。これらカラーの設定については，COLOR, CONSOLEの項を参照してください。



1.19.2 N80-BASICのカラー

N80-BASICのカラーはカラーコードで指定するものと，カラーナンバで指定するものと2種類あります。カラーコードはN-BASICと同じで，主にテキスト画面に対し色指定を行ない，色は8色まで使用できます。また，カラーナンバは高解像度グラフィックスに対して色指定するときに使用します。モノクロモード，アトリビュートカラーモードのときは2色まで（0，1），4色カラーモードのときは4色まで（0，1，2，3），指定できます。

後述するCMD SCREENによってこれらのモードが決定されます。

カラーコードで指定を行うもの

PSET

PRESET

LINE

PUT@

COLOR

CMD COLOR@

カラーナンバで指定を行うもの

CMD PSET

CMD PRESET

CMD LINE

CMD CIRCLE

CMD PUT@

CMD COLOR

詳しくはPC−8001MKⅡユーザーズマニュアルを参照してください。

1.20 ファイル

ファイルとは意味を持つ情報の集まりで，N$_{80}$(N)-BASICはフロッピィディスクやカセットテープとの入出力を行う時，ファイルを一つの単位として管理します。ファイルにはデータファイルとプログラムファイルの2つがあります。

さらにフロッピィディスク上のデータファイルを使用するにあたって，そのファイルにファイル番号を割り当て，ファイル番号でファイルの入力出力を行うため能率よくファイル操作を行うことができます。

1.20.1 ファイル名

〈ファイル名〉とはプログラムファイルやデータファイルに付ける名前で，これはユーザが指定します。 ファイル名を必要とするのは，カセットテープ,フロッピィディスクなど補助記憶装置との入出力を行う場合で，その他の場合は省略することができます。ファイル名は次のような書式をとります。

〈ファイル名〉〔.〕〔**〈拡張子〉**〕

最初の〈ファイル名〉は6文字，ピリオドに続く〈拡張子〉は，もしあれば3文字までの文字列より構成されます。もしそれぞれの文字数がそれ以下の場合には空いた部分に空白が埋められます。一般に〈ファイル名〉がファ

イルの名前を,〈拡張子〉がファイルの性質を表すようにしますが,ユーザによって自由に使うことができます。通常ファイル名と言う場合は〈拡張子〉まで含んだものを意味します。

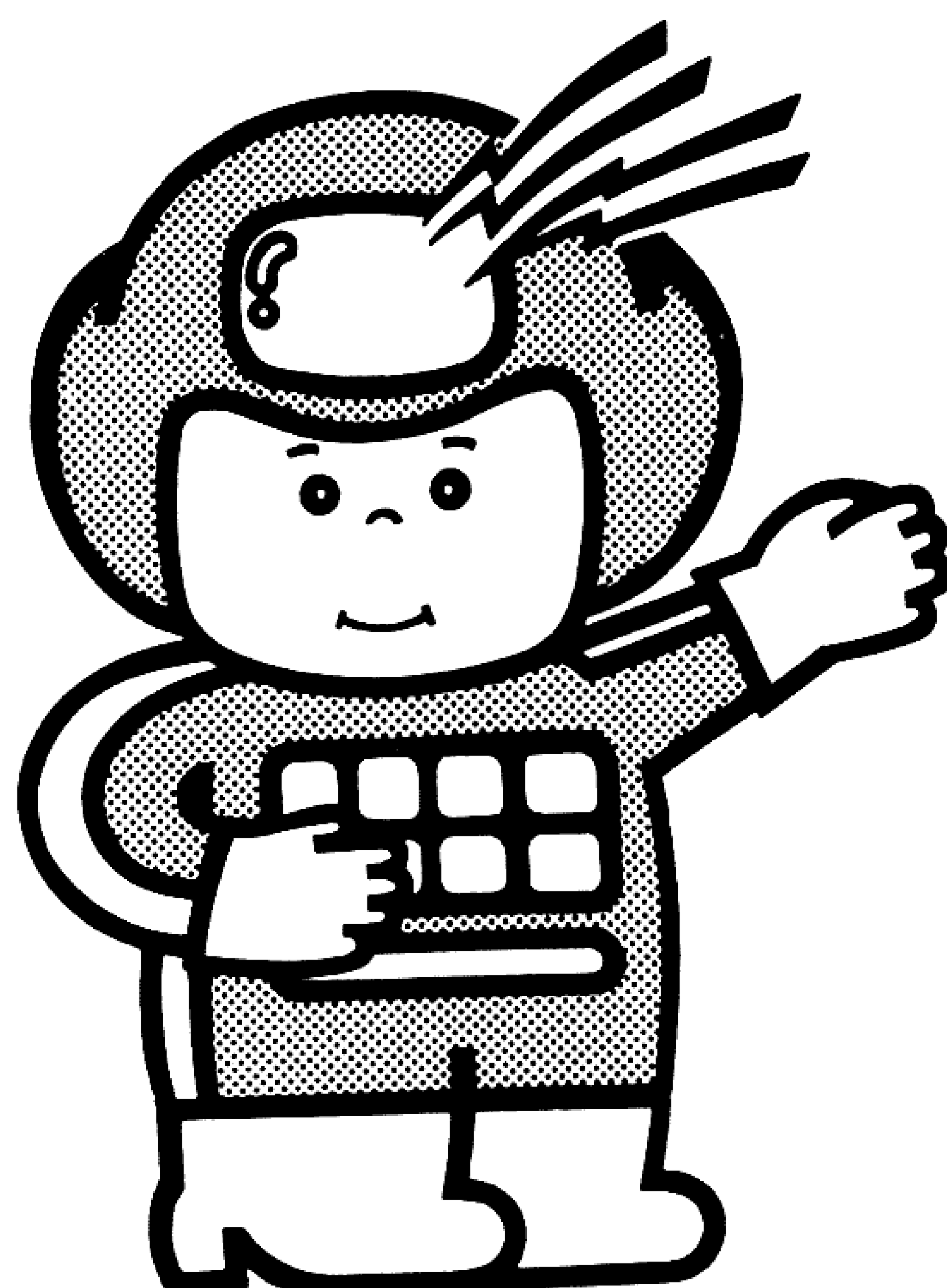
ただし，コロン(:)及びキャラクタコード0と255で表される文字を含むことは許されません。

また最初の〈ファイル名〉が6文字を越えて指定さた場合，先頭から6文字が〈ファイル名〉と，続く9文字までの3文字が〈拡張子〉と見なされ，それ以降の文字は無視されます。

〈ファイル名〉の先頭にドライブ番号を指定します。ドライブ番号と〈ファイル名〉はコロン(:)で区切られます。ドライブ番号が1の場合は省略することができます。

第2章

コマンド・ステートメント
関数・予約変数



2章のみかた

2章で，N80-BASIC, N-BASICのすべてのコマンド，ステートメント，関数，予約変数を解説します。ここにKILL というコマンドを解説したページをあげて，この中に使われている記号や表記方法の説明をします。

表記例

①ディスク ②N80 ③N ④一般コマンド

KILL [ディスク] [N80] [N] **一般コマンド**

⑤ 機能　フロッピィディスク上のファイルを削除します。

⑥ 書式　KILL <ファイル名>

⑦ 文例　KILL "2 : test"

⑧ 解説
- <ファイル名>で指定したファイルをフロッピィディスクから削除します。
- 削除するファイルはクローズされていなければなりません。もし，オープン状態のファイルを削除しようとすると "File already open" エラーが起ります。また，書き込み禁止属性の付けられたファイルを KILL で削除しようとすると "File write protected" エラーが起ります。この場合 SET で書き込み禁止属性を解除してから KILL を実行してください。
- KILL では1度に1つのファイルしか削除できません。
- KILL はすべてのディスクファイル(プログラム，ランダムファイル，シーケンシャルファイル)について使用できます。

⑨ 注意　ドライブ番号はファイル名の前に "n :" と指定します。省略されたときはドライブ1が指定されたとみなされます。したがって複数のドライブの中に同じファイル名で指定されたフロッピィディスクがある場合注意が必要です。

⑩ 参照　SET, ATTR$

⑪ サンプルプログラム

```
100 OPEN "test2" FOR OUTPUT AS #1
110 PRINT #1,"KILL statement test"
120 CLOSE
130 '
140 FILES:PRINT
150 KILL "test2"
160 FILES
```

① [ディスク] このマークがついているものは DISK-BASIC モードでないと使えません。マークのついていないものはDISK-BASICモード，ROM-BASICモードのどちらでも使えます。

② [N80] N80-BASICで使えることを示します。

③ [N] N-BASICで使えることを示します。

①〜③のマークの組み合わせによって以下のようなモードで使えることを示します。

N80 N	N80-BASIC, N-BASIC, N80DISK-BASIC, DISK-BASIC
N80	N80-BASIC, N80 DISK-BASIC
ディスク N80 N	N80 DISK-BASIC, DISK-BASIC
ディスク N80	N80DISK-BASIC
ディスク N	DISK-BASIC

④　機能別分類

BASICのコマンド・ステートメント・関数・予約変数のうちのどれであるか，どのような機能をもつものかを示します。

一般コマンド **一般ステートメント**	プログラムを作る，動作させる，停止させる，変数などの定義をするといった基本的な動作や，プログラムの流れのコントロール(ループや分岐)を行います。
入出力コマンド **入出力ステートメント** **入出力関数**	フロッピィディスクユニット，カセットテープレコーダ，画面，キーボード，プリンタ，スピーカ，RS-232Cポートなどの操作を行います。
グラフィックステートメント **グラフィック関数**	高解像度グラフィックス，低解像度グラフィックスに用います。
画面制御ステートメント **画面制御関数**	テキスト画面，グラフィックス画面のイニシャライズやコントロールに用います。
数値関数 **文字関数** **文字ステートメント**	算術演算と文字列の操作を行います。
特殊コマンド **特殊ステートメント** **特殊関数**	モニタやターミナルモードに入るときやエラー処理，機械語の操作などの特殊な働きをします。
予約変数 ………………………	BASICが専用に使っている変数です。

⑤ (**機　能**) 機能を簡単に示します。

⑥ (**書　式**) 記述方法の定義を行います。 以下の規則に従ってください。

1. <　>に囲まれていない文字や記号は，そのまま入力します。入力する文字は大文字でも小文字でもかまいません。

ただし，次の場合は，大文字，小文字を区別しなければなりません。

1）ダブルクォーテーションマーク（〝 ）で囲まれた文字列は必ず大文字・小文字を区別しなければなりません。

例）〝MARK2〟, 〝format.N80〟　（ファイル名）
　　SET1, 〝P〟　（属性文字）

2）数値を指数形式で表わす場合に用いるEとDは必ず大文字でなければなりません。

例）A!=−7.09E−06
　　A＃=1.094328564D5

2. カギカッコ〝<〟，〝>〟で囲まれた項目は，その定義内容をユーザが具体的に指定します。

次のような種類があります。

<数式>：　数値定数や数値変数を，単独であるいは演算子で結合したもの。
例）**1, 10, 1/SIN(A!), B2, C＊D**

<文字列>：　文字定数あるいは文字型変数。
例）**"ABC", ST$**

<式>：　**<数式>**または**<文字列>**

<機能>, <スイッチ>, <コード>：
0, 1, 2, 3などの整数で，コマンドやステートメントの機能を指定するもの。

<ファイル名>：　カセットテープやフロッピィディスク上のファイルを表わす名前で "ドライブ番号：ファイル名．拡張子"の形式を持つもの。
例）**"MERGE", "3 : backupN80"**

<ファイル番号>： ファイルをオープンするときに定義される。データの読み書きなどには，<ファイル名>でなく<ファイル番号>を用いる。カセットテープ上のデータファイルは,. オープンせずに使い，<ファイル番号>＝－1である。

<ドライブ番号>： フロッピィディスクユニットの各々のドライブに割り当てられる番号。

<カラーコード>　**MKII** が表示可能な 8 色の色につけられた番号。

0：黒　1：青　2：赤　3：マゼンタ
4：緑　5：シアン　6：黄　7：白

<カラーナンバ>： 高解像度グラフィックスにおいて使用可能な色につけられる番号。

(X, Y)　　：キャラクタ座標
(Hx, Hy)　：高解像度グラフィックス座標
(Lx, Ly)　：低解像度グラフィックス座標
STEP(x, y)：相対座標

3．角カッコ“〔”，“〕”で囲まれた項目は，オプションであり省略することができます。省略した場合，デフォルト値(BASICによって設定される値)または以前に指定した値が適用されます。

4．省略記号“…”の続く項目は，1行の許す長さ(255文字)の内で任意の回数繰り返すことができます。

例）**<定数>〔,<定数>…〕の場合**　0, 10, 15

5．△はスペースを表わします。

⑦ **(文　例)** 実際の入力の見本として簡単な例を示します。

⑧ **(解　説)** 命令の使用方法や詳しい機能とそれに関して注意しなければならない点などを説明します。

⑨ **(注　意)** 特に注意してほしい点を示します。

⑩ **(参　照)** 関連の深い項目を示します。

⑪ **(サンプルプログラム)** いくつかの文を交えたプログラム例とその実行結果を示します。

ABS　N80 N　数値関数

機能　**絶対値を与えます。**

書式　**ABS**(〈**数式**〉)

文例　**B＝ABS(−8)**

解説
- absoluteの意味で，〈数式〉の絶対値を与えます。
- ABS関数の演算結果は，〈数式〉に倍精度実数が含まれる場合は倍精度に，その他の場合は単精度となります。

サンプルプログラム

```
100 '-- 2ﾃﾝ ｶﾝ ﾉ ｷｮﾘ ｦ ﾓﾄﾒﾙ --
110 INPUT "ﾃﾝ 1 ﾉ ｻﾞﾋｮｳ ﾊ: x,y ";X1,Y1
120 INPUT "ﾃﾝ 2 ﾉ ｻﾞﾋｮｳ ﾊ: x,y ";X2,Y2
130 D=SQR(ABS(X1-X2)^2+ABS(Y1-Y2)^2)
140 PRINT:PRINT "ﾃﾝ1 ﾄ ﾃﾝ2 ﾉ ｷｮﾘ ﾊ ";D

run
ﾃﾝ 1 ﾉ ｻﾞﾋｮｳ ﾊ: x,y ? 0,0
ﾃﾝ 2 ﾉ ｻﾞﾋｮｳ ﾊ: x,y ? 60,80

ﾃﾝ1 ﾄ ﾃﾝ2 ﾉ ｷｮﾘ ﾊ  100
Ok
```

ASC　N80 N　文字関数

機能　**文字を対応するキャラクタコードに変換します。**

書式　**ASC(〈文字列〉)**

文例　**B=ASC("A")**

解説

- 〈文字列〉の最初の文字を，対応するキャラクタコードに変換します。
- 〈文字列〉にヌルストリングを与えたときには，"Illegal function call" エラーが起ります。

参照　CHR$，付録E キャラクタコード

サンプルプログラム

```
100 'ASC
110 '-- lower chr.  ->  upper chr. --
120 INPUT A$
130 IF A$="" THEN 120
140 B$=""
150 FOR I=1 TO LEN(A$)
160   D$=MID$(A$,I,1):D=ASC(D$)
170   IF D<97 OR D>122 THEN B$=B$+D$:GOTO 190
180   B$=B$+CHR$(D-32)
190 NEXT I
200 PRINT:PRINT B$
210 END

run
? nec pc-8001mk2

NEC PC-8001MK2
Ok

run
? ****  personal computer  ****

****  PERSONAL COMPUTER  ****
Ok
```

ATN　[N₈₀] [N]　数値関数

機能　**逆正接(アークタンジェント)を与えます。**

書式　**ATN(〈数式〉)**

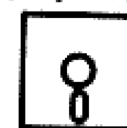
文例　**A=ATN (1)**

解説
- 〈数式〉の単位をラジアン(π/180×角度)としたときの逆正接(アークタンジェント)を与えます。
- 得られる値は−π/2からπ/2までの範囲となります。
- ATN 関数の演算は単精度で行われます。

サンプルプログラム

```
100 '-- arcTAN ｶﾗ arcSIN,arcCOS ｦ ﾓﾄﾒﾙ --
110 T=180/3.14159:U1$="#.###":U2$="##.####     ###.##"
120 INPUT "arcTAN";A
130 IF ABS(A)>1 THEN 120
140 IF ABS(A)=1 THEN AS=1.5708:AC=0:GOTO 170
150 AS=ATN(A/SQR(-A*A+1))
160 AC=-ATN(A/SQR(-A*A+1))+3.14159/2
170 DS=AS*T:DC=AC*T
180 PRINT "                     Rad.       Deg."
190 PRINT "arcCOS(";USING U1$;A;:PRINT ") =";USING U2$;AC;DC
200 PRINT "arcSIN(";USING U1$;A;:PRINT ") =";USING U2$;AS;DS
210 END

run
arcTAN? .5
                       Rad.       Deg.
arcCOS(0.500) = 1.0472      60.00
arcSIN(0.500) = 0.5236      30.00
Ok
```

ATTR$    入出力関数

(機　能) **指定したファイル，または，ドライブの属性を与えます。**

(書　式) **ATTR$** | (<ドライブ番号>) / (#<ファイル番号>) / (<ファイル名>) |

(文　例) **PRINT ATTR$ (1)**

(解　説) • 指定したドライブに入っているフロッピィディスク，<ファイル番号>によって指定されたオープンされているファイル，<ファイル名>で指定されたフロッピィディスク上のファイルの現在の属性を3文字の文字列で与えます。属性を表わす文字は次のような意味を持ちます。

"△△△"：属性は"なし"です。すなわち，通常のリードライトが可能な状態です。

"R△△"：書き込みのときに，リードアフターライト(書き込んだ内容とメモリ上の内容の比較チェック)を行います。

"△△P"：書き込みが禁止されています。

(参　照) SET

(サンプルプログラム)

```
100 '-- ATTRIBUTE CHECK  ----
110 INPUT 'file name: ';F$
120 IF MID$(F$,2,1)<>':' THEN 150
130 DR=VAL(F$):F1$=MID$(F$,3)
140 GOTO 160
150 DR=1:F1$=F$
160 AT$=ATTR$(DR)
170 PRINT 'drive ';STR$(DR);' attribute : ';:GOSUB 220
180 IF MID$(F$,3)='' THEN 210
190 AT$=ATTR$(F$)
200 PRINT F1$;' attribute : ';:GOSUB 220
210 END
220 A$='None'
230 IF AT$='R  ' THEN A$='Read after write'
240 IF AT$='  P' THEN A$='Write protect'
250 PRINT A$
260 RETURN
run
file name: ? test
drive  1 attribute : None
test attribute : Read after write
Ok
```

AUTO　N80 N　一般コマンド

機能　**行番号を自動的に発生させます。**

書式　**AUTO〔〈行番号〉〕〔,〈増分〉〕**

文例　**AUTO 100, 20**

解説

・　AUTO コマンドは，指定された〈行番号〉を最初の行番号とし，以後 **RETURN** キーの入力ごとに〈増分〉で増加された行番号を発生させます。

・　〈行番号〉，〈増分〉は，0 ～ 65529まで(〈増分〉は 1 ～)の整数であり，指定のないときは，それぞれ10と見なされます。

・　CTRL－C または，ストップキーを押すと，AUTO コマンドはその動作を中止し，BASIC のコマンドレベルに戻ります。この時，最後に発生させた行番号の行は格納されません。

・　AUTO コマンドの動作中において，既に表示されている行を，カーソルを移動して修正することはできません。AUTOコマンドを終了してから修正してください。

サンプルプログラム

```
auto 100,20
100 '-- test --
120 print "auto test"
140 end
160
Ok
list
100 '-- test --
120 PRINT "auto test"
140 END
Ok
```

BEEP  入出力ステートメント

機能 **内蔵スピーカによりブザーを鳴らします。**

書式 **BEEP〔〈スイッチ〉〕**

文例 **BEEP**

解説

- 内蔵スピーカから音を出したり，止めたりします。
- 〈スイッチ〉の値が1のときブザーは鳴りっぱなしとなり，〈スイッチ〉0で止まります。
- 〈スイッチ〉を省略した場合には，PRINT CHR$(7) を実行したのと同じで，一定時間(約0.5秒)ブザーを鳴らします。

参照 付録H コントロールコード

サンプルプログラム

```
100 '-- bell 1 --
110 FOR I=1 TO 50
120   BEEP 1
130   BEEP 0
140 NEXT I
150 GOSUB 280
160 '-- bell 2 --
170 FOR I=1 TO 30
180   FOR J=I TO 30
190     BEEP 1
200   NEXT J
210 BEEP 0
220 NEXT I
230 GOSUB 280
240 '
250 PRINT CHR$(7)
260 END
270 '-- wait routine --
280 FOR J=0 TO 200
290 NEXT J
300 RETURN
```

CDBL　N80 N　数値関数

機能　**整数値，単精度実数値を倍精度実数値に変換します。**

書式　**CDBL(<数式>)**

文例　**A＃＝CDBL(B!＊2)**

解説

- 〈数式〉の値を倍精度実数値に変換します。但し，型変換が行われるだけで有効桁数の変化はありません。
- 結果の値の精度は，変換する前の型と同じ(整数型なら整数部のみ，単精度実数型なら有効数字6桁)になります。

参照　CINT, CSNG

サンプルプログラム

```
100 'CDBL
110 '-- ｺﾞｻ ﾉ ﾅｲ ﾊﾞｲｾｲﾄﾞ ｼﾞｯｽｳ ﾍﾉ ﾍﾝｶﾝ --
120 A!=1.23456:GOSUB 160:GOSUB 220
130 A!=123456!:GOSUB 160:GOSUB 220
140 A!=1.23456E+08:GOSUB 160:GOSUB 220
150 END
160 A$=STR$(A):A1#=CDBL(A!)
170 D=INSTR(A$,".")
180 IF D=0 OR INSTR(A$,"E")<>0 THEN A2#=A1#:RETURN
190 D=LEN(A$)-D
200 A2#=INT(CDBL(A!*10^D+.5))/10^D
210 RETURN
220 PRINT A!,A1#;TAB(34);A2#
230 RETURN

run
 1.23456        1.234559893608093   1.23455980710003
 123456         123456              123456
 1.23456E+08    123456000           123456000
Ok
```

CHR$ N80 N 文字関数

機能 キャラクタコードを対応する文字に変換します。

書式 CHR$(〈数式〉)

文例 A$=CHR$(236)

解説
- 〈数式〉の値をキャラクタコードとし，対応する文字，またはコントロールコードに変換します。
- 〈数式〉は，０～255までの範囲でなければなりません。

参照 ASC, 付録Eキャラクタコード

サンプルプログラム

```
100 'ASC
110 '-- lower chr.  ->  upper chr. --
120 INPUT A$
130 IF A$="" THEN 120
140 B$=""
150 FOR I=1 TO LEN(A$)
160   D$=MID$(A$,I,1):D=ASC(D$)
170   IF D<97 OR D>122 THEN B$=B$+D$:GOTO 190
180   B$=B$+CHR$(D-32)
190 NEXT I
200 PRINT:PRINT B$
210 END

run
? nec pc-8001mk2

NEC PC-8001MK2
Ok

run
? ****  personal computer  ****

****  PERSONAL COMPUTER  ****
Ok
```

CINT　[N80] [N]　数値関数

機能　**単精度実数値，倍精度実数値を整数値に変換します。**

書式　**CINT(〈数式〉)**

文例　**A%=CINT(B# *2)**

解説

- 〈数式〉の値の小数部分を切り捨てて整数に変換します。
- 結果の値が－32768～32767の範囲にないときには，

"Overflow" エラーが起こります。

参照　CSNG, CDBL

サンプルプログラム

```
100 '-- ｼｮｳｽｳﾃﾝｲｶ ﾆｲ ﾉ ｹﾀ ﾃﾞﾉ 4ｼｬ5ﾆｭｳ --
110 A=1.23456:D=2:GOSUB 140:GOSUB 170
120 A=1.23456:D=3:GOSUB 140:GOSUB 170
130 END
140 IF A=INT(A) THEN AD=A:RETURN
150 AD=CINT(A*10^D+.5)/10^D
160 RETURN
170 PRINT A,AD
180 RETURN

run
 1.23456        1.23
 1.23456        1.235
Ok
```

CLEAR　N80 N　特殊ステートメント

(機能)　**変数の初期化およびメモリ領域の設定をします。**

(書式)　**CLEAR〔<ストリング領域の大きさ>〔,<メモリの上限>〕〕**

(文例)　**CLEAR 1000, &HDFFF**

(解説)

- すべての数値変数を0に，文字変数を“”(ヌルストリング)に初期化します。
- 〈ストリング領域の大きさ〉はBASICが文字列を処理するために用いるメモリ領域の大きさをバイト数で指定します。多くの文字列演算を行ったり大きな文字配列を用いる場合には，この領域を大きく指定しないと“Out of string space”エラーが起ります。リセット後の初期化された状態では，この領域は300に初期設定されます。
- 〈メモリの上限〉はBASICが使用するメモリの上限番地を指定します。その番地以後に置かれたデータや機械語プログラムは，BASICによって破壊されることはありません。
- CLEARはDEF (DEFFN, DEFUSR, DEFINTなど)によって定義あるいは指定された情報もすべて無効にします。

(サンプルプログラム)

```
100 '-- valuable clear --
110 DEF FNX(X)=X^2+X+5
120 A%=213
130 B=74.7543
140 C#=76346.43263654302#
150 D$="ABCDEFG"
160 DIM E(10)
170 FOR I=0 TO 10
180   E(I)=I
190 NEXT
200 PRINT
210 GOSUB 270:'-- DUMP --
220 CLEAR
230 GOSUB 270:'-- DUMP --
240 END
250 '
260 '-- DUMP --
270 PRINT "a%=";A%,"b!=";B,"C#=";C#
280 PRINT "a$=";CHR$(&H22);D$;CHR$(&H22)
290 PRINT "e(x)=",
300 FOR I=0 TO 10
310 PRINT E(I);
```

```
320 NEXT I:PRINT
330 PRINT "FNX(X)=";FNX(10)
340 PRINT :PRINT
350 RETURN
```

```
run

a%= 213         b!= 74.7543    C#= 76346.43263654302
a$='ABCDEFG'
e(x)=            0  1  2  3  4  5  6  7  8  9  10
FNX(X)= 115


a%= 0           b!= 0          C#= 0
a$=''
e(x)=            0  0  0  0  0  0  0  0  0  0  0
FNX(X)=
Undefined user function in 330
Ok
```

CLOAD　N80 N　入出力コマンド

機能　**カセットテープからメモリへプログラムをロードします。**

書式　**CLOAD〈ファイル名〉**

文例　**CLOAD "DEMO"**

解説

- CLOAD は，カセットテープから指定した〈ファイル名〉のプログラムをメモリへロードする命令です。
- 〈ファイル名〉は 6 文字以内で指定してください。
- CLOAD コマンドの後，カセットテープから指定した〈ファイル名〉を見つけると

　　Found：〈ファイル名〉

と表示され，プログラムをロードします。そのとき画面右上にアスタリスク(＊)が点滅します。

- もし指定した〈ファイル名〉と異った〈ファイル名〉を見つけると

　　Skip：〈ファイル名〉

と表示して，目的の〈ファイル名〉を探し続けます。

- プログラムをロードするのに成功すれば画面に "Ok" と表示し，失敗すれば "Tape read ERROR" と表示してからコマンドレベルに戻ります。

参照　PC-8001MKIIユーザーズマニュアル，CSAVE

サンプルプログラム

```
cload"baka"
Skip:clock
Found:baka
Ok
```

CLOAD? [N80] [N] 入出力コマンド

(機能) **メモリ上のプログラムとテープ上のプログラムとの比較をします。**

(書式) **CLOAD?<ファイル名>**

(文例) **CLOAD?"DEMO"**

(解説)
- CLOAD?は現在メモリ上にあるプログラムと<ファイル名>で指定されたテープ上のプログラムを比較(ベリファイ)し，すべての内容が等しければ"Ok"と表示し，そうでなければ"Bad"と表示してからコマンドレベルに戻ります。
- CLOAD?はメモリ上のプログラムがミスなくテープ上にセーブされているかを確認するためのものであり通常はCSAVEに引き続いて実行します。この場合プログラムを実行する前にCLOAD?を行います。

(注意) CLOAD?はCSAVEの直後に行ってください。CSAVEのあと1度プログラムを実行するとCLOAD?でプログラムの比較を行っても等しくならないことがあります。

(参照) PC-8001 MKIIユーザーズマニュアル，CLOAD, CSAVE

(サンプルプログラム)
```
cload?"sumo"
Found:sumo
Ok
```

CLOSE ディスク   入出力ステートメント

機能 **ファイルを閉じます。**

書式 **CLOSE** 〔〔＃〕<**ファイル番号**>〔,〔＃〕<**ファイル 番号**>〕…〕

文例 **CLOSE**

解説 • <ファイル番号>に対応するファイルを閉じます。以後，CLOSE によって指定された<ファイル番号>は，異なるファイルを開くために再び利用することができます。また，閉じられたファイルは，同じあるいは異なったファイル番号によって再び開くことができます。

• CLOSE では，<ファイル番号>を複数指定することにより一度に複数のファイルを閉じることができます。<ファイル番号>が省略された場合には，そのとき開いているファイルをすべて閉じます。

• 閉じているファイルに対して，データの入出力を行うことはできません。

• ファイルが出力用にオープンされていた場合には，必ず CLOSE を実行しなければなりません。

• END, NEW を実行すると自動的にすべてのファイルを閉じます。STOP はファイルを閉じる作業は行いませんので注意してください。

注意 N80DISK-BASICにおいて，ファイルを開いたままフロッピィディスクをとり出すと，フロッピィディスクの配置表(FAT)をこわすおそれがあります。したがって，ファイルをオープンした場合，必ず CLOSE ですべてのファイルを閉じてからフロッピィディスクをとり出してください。

サンプルプログラム

```
100 OPEN "test1" FOR OUTPUT AS #1
110 OPEN "test2" FOR OUTPUT AS #2
120 PRINT #1,"file data 1"
130 CLOSE #1
140 PRINT #2,"file data 2"
150 CLOSE #2
```

CMD BLOAD  入出力コマンド

機能 **機械語プログラムをロードします。**

書式 **CMD BLOAD〈ファイル名〉〔,〈ロードアドレス〉〕〔,R〕**

文例 **CMD BLOAD "DEMO. bin" ,R**

解説
- 〈ファイル名〉によって指定された，フロッピィディスク上の機械語プログラムファイルをメモリにロードします。ただし，機械語プログラムファイルはCMD BSAVEで作成されたものでなければなりません。
- 〈ロードアドレス〉が省略された場合，CMD BSAVEによりファイルをセーブする際に指定したものと同じになります。また,〈ロードアドレス〉が指定されるとその番地からロードし始めます。このとき，セーブされた番地と異なる番地にロードされても，実行可能(リロケータブル)になっていなければなりません。
- Rオプションを指定すると，プログラムをロード後，セーブの際に指定された開始番地から，プログラムの実行を直ちに開始します。この時，既に開かれているファイルはその状態を保持します。〈ロードアドレス〉が指定されている場合は，実行開始番地も〈ロードアドレス〉の番地になります。

注意 カセットテープを対象としたCMD BLOAD命令は，実行できません。

参照 CMD BSAVE

サンプルプログラム

```
run

cmd bload"MOOUSEBIN"
Ok
cmd bload"PANNELBIN",&h9000
Ok
```

CMD BSAVE　ディスク N80　入出力コマンド

機　能　**機械語プログラムをセーブします。**

書　式　**CMD BSAVE〈ファイル名〉, 〈セーブアドレス〉, 〈長さ〉**

文　例　**CMD BSAVE "DEMO. BIN", &HE000, &H40**

解　説

- メモリ上に置かれている機械語プログラムを，指定された〈ファイル名〉でフロッピィディスクへセーブします。
- 〈セーブアドレス〉で指定された番地から，〈長さ〉バイトの内容が，機械語プログラムとしてセーブされます。
- 〈セーブアドレス〉を実行開始番地としておくとCMD LOADにより，機械語プログラムのロード(と実行)が行えます。

注　意　カセットテープを対象としたCMD BSAVEは，実行できません。

参　照　CMD BLOAD

サンプルプログラム

```
cmd bsave"MOUSEBIN",&hb600,&h3f
Ok
```

CMD CIRCLE　N80　グラフィックステートメント

(機　能)　円(楕円)を描きます。

(書　式)　CMD CIRCLE |(Hx, Hy)／STEP(x, y)|, 〈半径〉〔,〈カラーナンバ〉〕〔,〈開始角度〉〕〔,〈終了角度〉〕〔,〈比率〉〕

(文　例)　CMD CIRCLE (90, 90), 40, 1

(解　説)

• ビュー座標の (Hx,Hy) を中心とし，〈半径〉で指定される大きさの円を描きます。

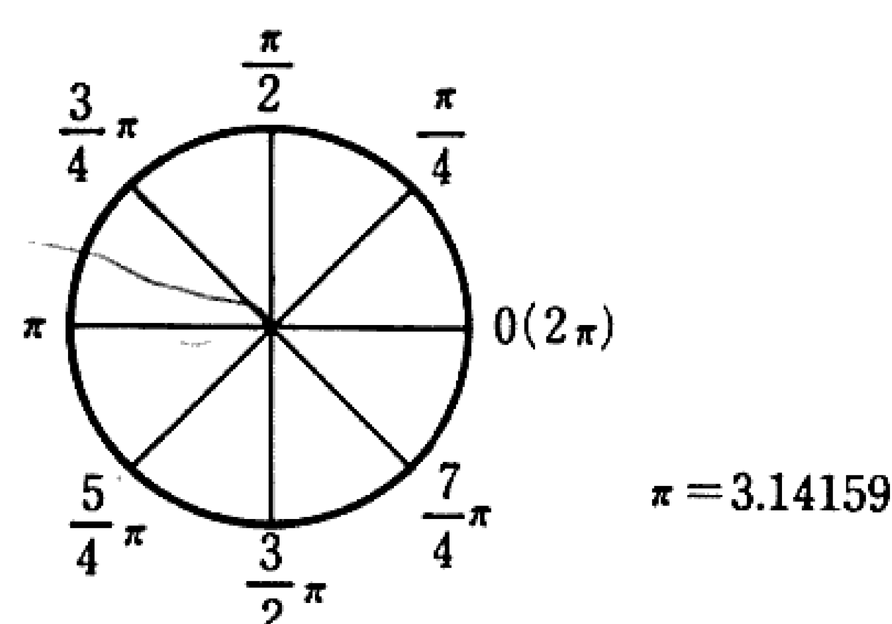
• 〈半径〉は，1ピクセルを単位としたときの長さです。また，〈半径〉が負であった場合は，その絶対値をとります。

• 〈カラーナンバ〉を指定すると，指定されたカラーナンバの色で円を描きます。省略された場合，CMD COLOR で指定されているフォアグラウンドカラーナンバで円を描きます。

• CMD CIRCLE は通常円を描きますが，〈開始角度〉，〈終了角度〉を指定すると指定された角度の範囲内のみに円弧を描きます。但し，角度は，真円を基準としたときの値です。したがって楕円で円弧を描く場合，注意が必要です。

• また，〈開始角度〉，〈終了角度〉が負であった場合，その絶対値をとり，正にした角度が用いられます。そのとき中心から半径が描かれ，〈開始角度〉，〈終了角度〉共に負であった場合，扇形を描くことができます。

• 〈開始角度〉，〈終了角度〉は，－2πから＋2πの範囲内にないと "Illegal function call" エラーが起こります。



• 〈比率〉は，(垂直方向の半径)／(水平方向の半径)で指定します。〈比率〉が省略された場合には，640×200ピクセルのモードでは0.5，320×200ピクセルのモードでは1.0が採用されます。

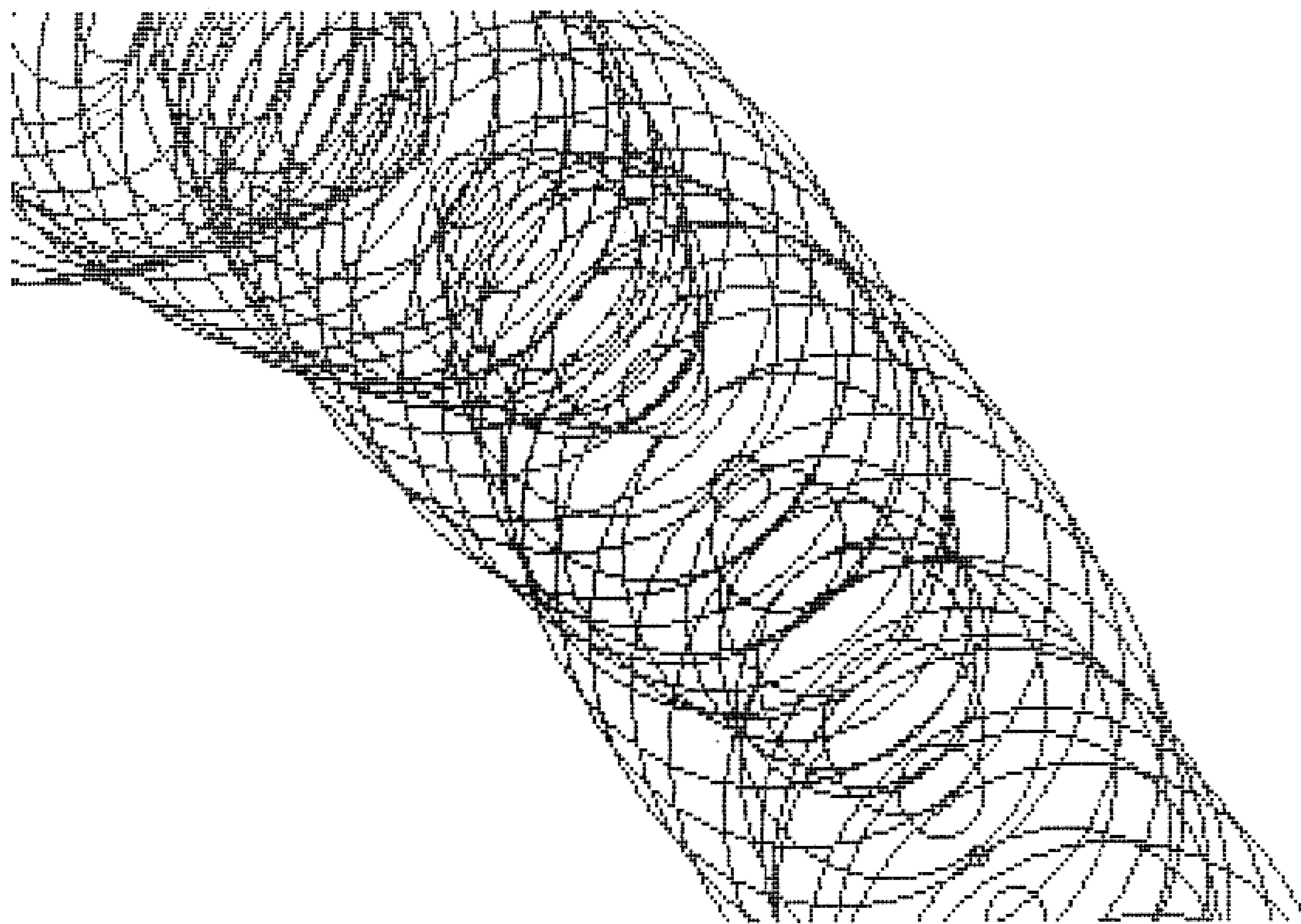
• 320×200ピクセルのモードで1未満の〈比率〉を指定したとき，〈半径〉は水平方向の半径が基準になります。また，1以上の〈比率〉を指定したときには，垂直方向の半径が基準となります。

• 640×200ピクセルのモードで1未満の〈比率〉を指定したとき，320×200ピクセルのモードと同様に〈半径〉は水平方向の半径が基準になります。また1以上の〈比率〉を指定したとき，〈比率〉が1のときの垂直方向の〈半径〉が基準になります。

• 中心座標 (Hx,Hy) はビューポートの外に指定することができます。しかし，極端に大きな円の一部(円弧)を表示させた場合，動作は保障されません。

サンプルプログラム

```
100 CMD SCREEN 2,0
110 CMD COLOR 3,0
120 CMD CLS 3
130 FOR I=0 TO 100
140 CMD CIRCLE(I*3,SIN(I/639)*20*I),COS(I)*60
150 NEXT I
160 END
```



CMD CLS　N80　画面制御ステートメント

機能　画面をクリアします。

書式　CMD CLS〔〈機能〉〕

文例　CMD CLS 3

解説

• 〈機能〉は1，2，3の値をとり，それぞれ次のように働きます。省略されたときは1が選択されます。

1: テキスト画面をクリアします。 HOME CLR キーの入力，PRINT CHR$(12)；の実行と全く同じ動作をします。

2: ビューポート内のグラフィック画面を CMD COLOR で指定されたバックグラウンドカラーナンバでクリアします。

3: テキスト画面とグラフィック画面の両方をクリアします。

• グラフィック画面をクリアしたとき LP(Last referenced point)はビューポートの左上の頂点に移動します。またテキスト画面をクリアした場合カーソルはホームポジション(画面の左上)に移動します。

参照　CMD SCREEN, CMD COLOR, CMD VIEW, COLOR

サンプルプログラム

```
10 CMD SCREEN 2
20 CMD CLS3
30 FOR Y=0 TO 99
40 X=Y*16/5
50 CMD VIEW (X,Y)-(319-X,199-Y)
60 CMD COLOR ,RND(1)*4
70 CMD CLS 2
80 NEXT
90 FOR I=0 TO RND(1)*100+100
100 FOR M=3 TO 2 STEP -1
110 CMD SCREEN M,,RND(1)*8
120 NEXT
130 NEXT
140 CMD COLOR ,0
150 END
```

CMD COLOR　N80　グラフィックステートメント

機能　グラフィック画面のフォアグラウンドカラー，バックグラウンドカラーを指定します。

書式　CMD COLOR〔〈フォアグラウンドカラーナンバ〉〕〔,〈バックグラウンドカラーナンバ〉〕

文例　CMD COLOR 1, 0

解説

- グラフィック画面のフォアグラウンド，バックグラウンドの色をカラーナンバで指定します。
- 〈フォアグラウンドカラーナンバ〉とは，グラフィック画面に点や線を表示するときに使われる色のことです。N80-BASICの種々の高解像度グラフィック命令(CMD PSET, CMD LINE, CMD CIRCLEなど)で〈カラーナンバ〉の指定をしなかった場合，このフォアグラウンドカラーナンバの色が採用されます。
- 〈バックグラウンドカラーナンバ〉とは，グラフィック画面の地の色のことで，CMD COLOR実行後，CMD CLSによってグラフィック画面をクリアすると，指定したカラーナンバの色によって画面が塗り変えられます。また，CMD PRESETを〈カラーナンバ〉の指定なしで実行すると，〈バックグラウンドカラーナンバ〉の色が採用されます。
- 次にカラーナンバと色の関係を示します。

グラフィックモード ／ カラーナンバ	モノクロモード	アトリビュートカラーモード	4色カラーモード0	4色カラーモード1
0	黒	選択色(黒)	黒	青
1	選択色(白)	テキスト画面の色に依存します。	赤	マゼンタ
2	———	———	緑	シアン
3	———	———	選択色(青)	選択色(黒)

- 選択色とはCMD SCREENで指定する〈選択色のカラーコード〉のことで，(　)の色は CMD SCREENで〈選択色のカラーコード〉が省略されたときのものです。これらの指定のしかたはCMD SCREENを参照してください。
- 〈カラーナンバ〉はモノクロモード，アトリビュートカラーモードのときは0あるいは1を指定しますが，0以外の数値が指定されるとすべて1とみなします。また，4色カラーモード0，1のとき0～4までの値を指定しますが，その範囲外の数値は4で割ったあまりが採用されます。

サンプルプログラム

```
100 ' CMD COLOR
110 WIDTH 40,25
120 FOR S=2 TO 3
130  CMD SCREEN S,0
140    FOR I=0 TO 3
150      FOR J=0 TO 3
160        CMD COLOR I,J
170        CMD CLS 3
180        LOCATE 11,12
190        PRINT "cmd  color";I;",";J
200        CMD CIRCLE(150,100),70
210      NEXT J
220     NEXT I
230 NEXT S
240 END
```

CMD COLOR@ [N80] 画面制御ステートメント

機能 **テキスト画面に書かれた文字などに色を設定します。**

書式 **CMD COLOR@ (X_1, Y_1)−(X_2, Y_2)**

文例 **CMD COLOR@ (0, 0)−(40, 19), 3**

解説

- テキスト画面のキャラクタ座標の2点(X_1, Y_1), (X_2, Y_2)を対角とする四角形の領域に書かれている, 文字やグラフィックキャラクタに色を付けます。また, CMD SCREEN で〈グラフィックモード〉が1 (640×200ピクセル, アトリビュートカラーモード)のときフォアグラウンドカラーに色を設定することができます。ただし, このときの CONSOLE の〈カラー／白黒スイッチ〉は1 (カラーモード)でなければなりません。
- キャラクタ座標の範囲は次のようになります。

X_1,X_2= 0 〜 (〈桁数〉− 1)
Y_1,Y_2= 0 〜 (〈桁数〉− 1)

- なお, 〈桁数〉, 〈行数〉は現在 WIDTH 文で指定される値となります。

注意 CONSOLE で(カラーモード)を指定すると, テキスト画面はグラフィックス画面の左右の両端に重ならないので, グラフィックス画面の全領域に色をつけることはできません。
CMD COLOR@によって設定された領域の上に新しく文字などを書いた場合, 書かれた文字はこの命令の影響を受けません。また, 低解像度グラフィックで描かれている画面に CMD COLOR@を実行すると画面が乱れます。

参照 PC-8001MK II ユーザーズマニュアル

サンプルプログラム

```
100 WIDTH 80,25:CONSOLE 0,25,0,1
110 CMD CLS 3
```

```
120 CMD SCREEN 1,,0:CL=0
130 FOR J=0 TO 20 STEP 5
140   FOR I=0 TO 72 STEP 6
150     CMD COLOR@(I,J)-(I+6,J+4),CL
160     CL=CL+1:IF CL=8 THEN CL=0
170   NEXT I
180   '
190 NEXT J
200 FOR I=0 TO 639 STEP 2
210   CMD LINE(I,0)-(I,199),1
220 NEXT I
230 FOR I=0 TO 8
240   IF I=4 THEN 270
250   CMD SCREEN 1,,I
260   FOR J=1 TO 1000:NEXT J
270 NEXT I
280 IF INKEY$="" THEN 280
290 CMD SCREEN 0,,7:CMD CLS 3
```

CMD COPY　N80　入出力ステートメント

機　能　**画面に表示されている文字，図形をプリンタに出力します。**

書　式　**CMD COPY〔〈機能〉〕**

文　例　**CMD COPY 3**

解　説　• 〈機能〉は，1から5までの値をとり，次のように働きます。省略されたときは，3が選択されます。

1：テキスト画面のみをプリンタにコピーします。

2：グラフィック画面のみをプリンタにコピーします。

3：テキスト画面，グラフィック画面の両方をプリンタにコピーします。

4：グラフィック画面のみをプリンタにコピーします。640×200ピクセルのモードで出力された漢字(グラフィックパターン)はこのモードで出力すると縦方向が2分の1に縮小されて，見やすくなります。

5：テキスト画面，グラフィック画面の両方をプリンタにコピーします。4と同じように640×200ピクセルのモードで出力されたパターンは縦方向が2分の1に縮小されます。

注　意　1) CMD COPY文は PC-8023-C専用ですが，グラフィック画面のコピーにおいて，画面に表示されたものと，プリンタにコピーされたものとでは いくらかイメージが異なります。 また，PC-8821/22も使用することができます。しかし， PC-8023-C と同じようにプリンタの機械的な制約により グラフィック画面のコピーは 画面に表示されているものと プリンタにコピーされたものとでは形が異なります。

2） CMD COPY では，低解像度グラフィックスをプリンタにコピーすることはできません。

サンプルプログラム

```
100 'CMD COPY
110 'THis program cannot
120 '     run without PC-8023-C or PC-8021/22
130 '
140 CMD SCREEN 2,0,1
150 CMD CLS 3
160 '
170 FOR I=1 TO 20
180   X=RND(1)*37
190   Y=RND(1)*17
200   LOCATE X,Y:PRINT "MK2"
210   CMD CIRCLE((X+1)*8,Y*10),30,X*Y/4
220 NEXT I
230 '
240 FOR I=1 TO 5
250  CMD COPY I
260 NEXT I
270 '
280 END
```

CMD GET@ [N80] グラフィックステートメント

(機能) **グラフィック画面のグラフィックパターンを配列に読み込みます。**

(書式) **CMD GET〔@〕(Hx1, Hy1)−｜(Hx2, Hy2) / STEP(x, y)｜,〈配列変数名〉〔(〈要素ナンバ〉)〕**

(文例) **CMD GET〔@〕(0, 0)−(16, 16), G%**

(解説)

• 画面上のビュー座標の2点(Hx1, Hy1)と(Hx2, Hy2)を対角とする四角形の領域を,〈配列変数名〉で指定された配列変数に読み込みます。なお，ビュー座標とは，ビューポート内に展開される座標系(1章・ディスプレイ画面の座標系を参照)のことです。2つ目の座標指定には相対座標による指定が許されています。指定された座標が，ビューポートの範囲内にないときはエラーとなります。

• 〈配列変数名〉は，ユーザがグラフィックパターンを読み込むために用意する整数型の配列変数の名前です。またこの〈要素ナンバ〉は，グラフィックパターンを配列変数に読み込むときに，どの要素から格納し始めるかの指定です。1つの配列変数に対して複数のグラフィックデータを格納する場合に使うことができます。〈要素ナンバ〉を省略したときは配列の最初から格納します。

• GET@に先だってこの配列変数に必要な大きさをDIMで確保しなければなりません。この配列変数の大きさは，読み込む領域の大きさ，グラフィックモードによって異なります。

• 1つの配列に1つのパターンしか読み込まない場合は，次の計算式で求めることができます。複数パターンの場合は，それぞれのパターンについて計算し，その分確保しなければ

なりません。

〈必要なバイト数〉=((〈横のドット数〉*M+7)¥8)
*〈縦のドット数〉+4

M=1 — モノクロードモード，アトリビュートカラーモードのとき

M=2 — 4色カラーモード0，1のとき

〈添字の値〉=〈必要なバイト数〉¥2

- このGET@文は後述するPUT@文と密接な関係にあり，通常ペアで使用されます。
- GET@命令実行後，LPは(Hx_2,Hy_2)に移されます。

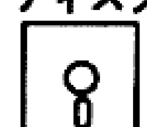
注意 配列変数は整数型の配列かつ一次元配列でなければなりません。

参照 CMD PUT@

サンプルプログラム

```
100 'CMD GET@ , CMD PUT@
110 CMD SCREEN 2,0
120 CMD CLS 3
130 XD=40:YD=40
140 BYTE=((XD+7)¥8)*YD*3+4
150 FACT=BYTE¥2+1
160 DIM G%(FACT)
170 'CMD GET@
180 CMD LINE(0,0)-STEP(XD-1,YD-1),1,B
190 CMD CIRCLE(XD/2-1,YD/2-1),YD/2-5 ,2
200 CMD GET(0,0)-STEP(XD-1,YD-1),G%
210 ' CMD PUT@
220 FOR X=0 TO 250 STEP 50
230 CMD PUT (X,100),G%
240 NEXT X
250 END


a=usr(0)
```

CMD INIT  N80 特殊コマンド

(機能) **フロッピィディスクのイニシャライズをします。**

(書式) **CMD INIT〔〈機能〉〕**

(文例) **CMD INIT 0**

(解説)

- N_{80} DISK－BASIC を使用している途中でウォームスタートをした場合，その後，フロッピィディスクユニットのイニシャライズをするために用います。
- 〈機能〉は 0 から 2 までの値をとり，省略された場合は 0 と解釈されます。システムディスクとディスクユニットとの組みあわせによって以下のように使い分けなければなりません。

0：最も一般的なディスクユニットのイニシャライズを行います。システムディスクとディスクユニットの関係が次のような場合です。

PC-8087	PC-8881 PC-8881(1)
PC-8037-1W	PC-8031-1W
PC-8037-2W	PC-8031-2W PC-80S31

1：ディスクユニットを片面モードにイニシャライズします。特に〈機能〉を 1 にする必要があるのは，次の組み合わせのように両面用のディスクユニットを片面モードで使用する場合に限られます。

PC-8037-1W	PC-8031-2W PC-80S31

片面用のディスクユニットを，片面用のシステムディスクで使用している場合は，CMD INIT 1 と CMD INIT 0 は同じ動作をします。

2:　ディスクユニットを両面モードにイニシャライズします。CMD INIT 0と同じ動作をしますから，通常はこのCMD INIT 2を使う必要はありません。

注　意

• 両面用のディスクユニットを両面モードで使用しているときにCMD INIT 1は，絶対に実行させないでください。実行させた場合は，ディスクユニットが片面モードになってしまい N80DISK-BASICが正しく動かなくなります。

• DISK-BASICの使用中に，ウォームスタートをした場合 CMD INITに相当する命令が用意されていません。この場合のフロッピィディスクユニットのイニシャライズ方法は，PC-8001MKIIユーザーズマニュアル第3章をごらんください。

サンプルプログラム

```
cmd init 0
Ok
cmd init 1
Ok
cmd init 2
Ok
```

CMD LINE　N80　グラフィックステートメント

(機能) 指定された2点間に直線を引きます。

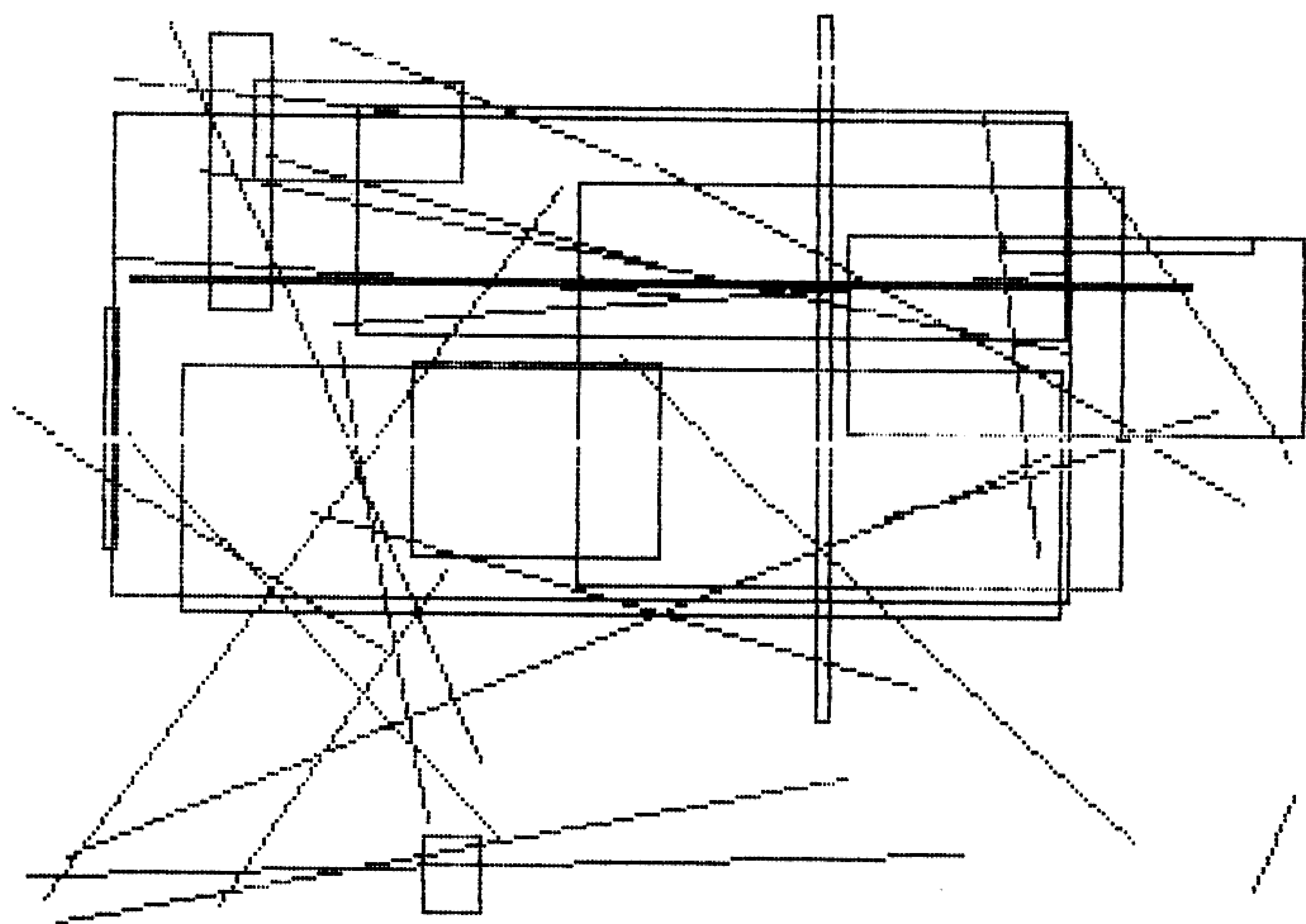
(書式) CMD LINE [{(Hx1, Hy1) / (Hx2, Hy2)}] − {(x1, y1) / STEP(x2, y2)} [, 〈カラーナンバ〉] [, {B / BF}]

(文例) CMDLINE(100, 100)−(200, 150), 1, B

(解説)
- ビュー座標系の2点(Hx1, Hy1)(Hx2, Hy2)を結ぶ直線を描きます。(Hx1, Hy1)が省略された場合，LPの値をとります。STEPを付けると相対座標による指定となります。(1章・座標の指定の仕方を参照)
- 引く線の色は〈カラーナンバ〉により指定し，もしこれが省略された場合，現在CMD COLOR文によって設定されている，グラフィック画面の〈フォアグラウンドカラーナンバ〉が採用されます。
- 次のパラメータには，BかBFのどちらかを指定することができます。BはBoxの意味で，(Hx1, Hy1)と(Hx2, Hy2)の2点を対角とする長方形を描き，BFはBox Fillの意味でその長方形を塗りつぶします。

(サンプルプログラム)

```
100 CMD SCREEN 2,0,1
110 CMD COLOR 2,0
120 CMD CLS 3
140   X1=RND(1)*319:Y1=RND(1)*199
150   X2=RND(1)*319:Y2=RND(1)*199
160   CL=RND(1)*3+1
170   MD=INT(RND(1)*2+1)
180   ON MD GOTO 190,200
190   CMD LINE(X1,Y1)-(X2,Y2),CL   :GOTO 260
200   CMD LINE(X1,Y1)-(X2,Y2),CL,B
260 IF INKEY$="" THEN 140
270 END
```



CMD PAINT　N80　グラフィックステートメント

機能　指定された境界色で囲まれた領域を,指定された色で塗ります。

書式　PAINT |(Hx, Hy) / STEP(x, y)| 〔, 〈領域色〉〔, 〈境界色〉〕〕

文例　PAINT(50, 50), 0, 1

解説

- (Hx, Hy)で塗り始めるビュー座標を指定します。
- 〈境界色〉で囲まれた領域を，指定された〈領域色〉で塗りつぶします。STEP を付けると相対座標となります(1章・座標の指定の仕方参照)。
- 〈領域色〉，〈境界色〉はともにカラーナンバで指定を行います。〈領域色〉が省略された場合，現在の CMD COLOR で指定されたフォアグラウンドカラーナンバが用いられます。〈境界色〉が省略された場合は，〈領域色〉の指定と同じカラーナンバが〈境界色〉として用いられます。
- (Hx, Hy)で指定された座標が，既に指定された境界色と同じ色であった場合には，CMD PAINT は何の画面操作も行いません。
- ビューポートの境界は CMD PAINT 動作の境界とみなされます。また，(Hx, Hy)がビューポートの範囲外に指定されてもエラーは起らず何の動作もしません。

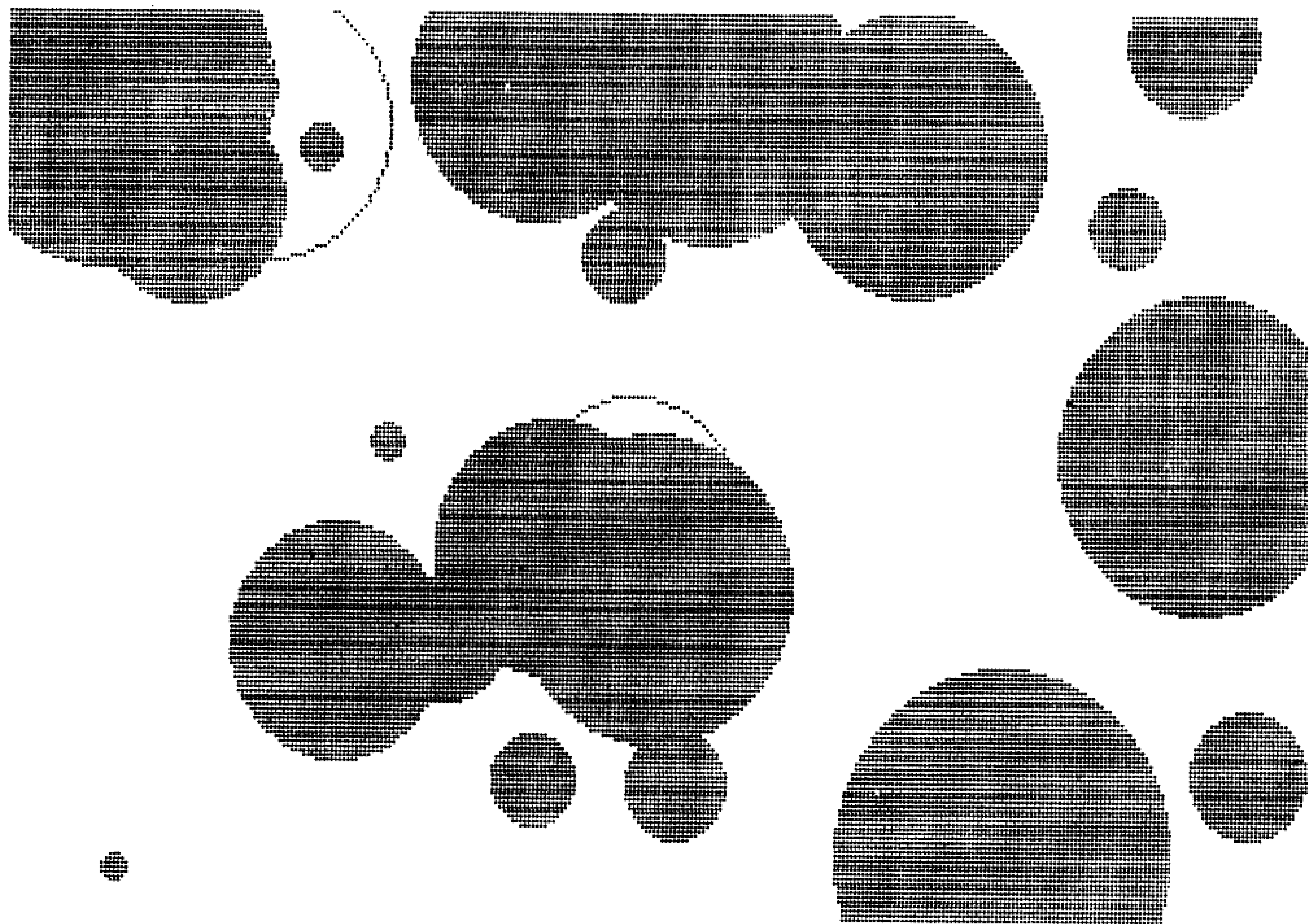
注意　複雑な図形を PAINT するとき，“Out of memory” エラーが起きることがあります。

参照　CMD COLOR, CMD VIEW

サンプルプログラム

```
100 'CMD PAINT
110 CMD SCREEN 2,0,1
120 CMD CLS 3
130 X=INT(RND(1)*319):Y=INT(RND(1)*199)
140 R=RND(1)*50:C=RND(1)*3+1
150 CMD CIRCLE (X,Y),R,C
```

```
160 CMD PAINT (X,Y),C
170 GOTO 130
```



CMD POINT　N80　グラフィックステートメント

(機　能)　**Last Referenced Point (LP) を変更します。**

(書　式)　**CMD POINT** | (Hx, Hy) / STEP(x, y) |

(文　例)　**CMD POINT (10, 30)**

(解　説)　• N80-BASICは，常に最後の高解像度グラフィック操作の行われた座標を覚えています。この座標は，相対座標により座標指定を行うときの基点となります。

• CMD POINT では，その基点をビュー座標で設定します。したがって次の例１)と２)は同じ結果となります。

1)　CMD LINE (40,100)−(90,50),1,BF

2)　CMD POINT (40,100)

　　CMD LINE−STEP (50,−50),1,BF

• STEP をつけると相対座標による指定となります。(1章・座標の指定の仕方参照)

(参　照)　STATUS POINT

(サンプルプログラム)

```
100 'CMD POINT
110 CMD SCREEN 2,,5:CMD CLS 3
120 FOR I=0 TO 35
130   CMD POINT STEP(8,3)
140   GOSUB 160
150 NEXT I:END
160 'triangle
170 CMD LINE -STEP(-50,50),1
180 CMD LINE -STEP(100,0),2
190 CMD LINE -STEP(-50,-50),3
200 RETURN
```

CMD PRESET  グラフィックステートメント

(機 能) グラフィック画面上の任意のピクセルをリセットします。

(書 式) CMD PRESET |(Hx, Hy) / STEP(x, y)| 〔, 〈カラーナンバ〉〕

(文 例) CMD PRESET (30, 40)

(解 説)

- ビュー座標の(Hx, Hy)で指定されるピクセルをリセットします。いいかえれば、現在CMD COLORによって設定されているバックグラウンドカラーで点を塗り変えます。
- 〈カラーナンバ〉を指定した場合は，CMD PSETと全く同じに機能し，そのパレット番号の〈カラーナンバ〉でピクセルをセットします。
- 座標値にSTEPを付けた場合は相対座標による指定となります(1章・座標の指定の仕方参照)。
- CMD PRESETの実行後，LPは(Hx,Hy)に設定されます。

(サンプルプログラム)

```
100 'CMD PRESET
110 CMD SCREEN 2,0
120 CMD COLOR 3,0
130 CMD CLS 2
140 CMD PAINT (0,0),3
150 CMD LINE(0,100)-(319,100),1
160 CMD LINE(50,0)-(50,200),1
170 'CIRCLE
180 P=3.14159
190 FOR T=0 TO 2*P STEP 2*P/100
200    X=50+50*COS(T):Y=100-50*SIN(T)
210   CMD PRESET(X,Y)
220   X=50+40*T
230   CMD PRESET (X,Y)
240 NEXT T
250 END
```

CMD PSET　N80　グラフィックステートメント

機　能　グラフィック画面上の任意の位置にピクセルをセットします。

書　式　CMD PSET {(Hx, Hy) | STEP(x, y)} [, 〈カラーナンバ〉]

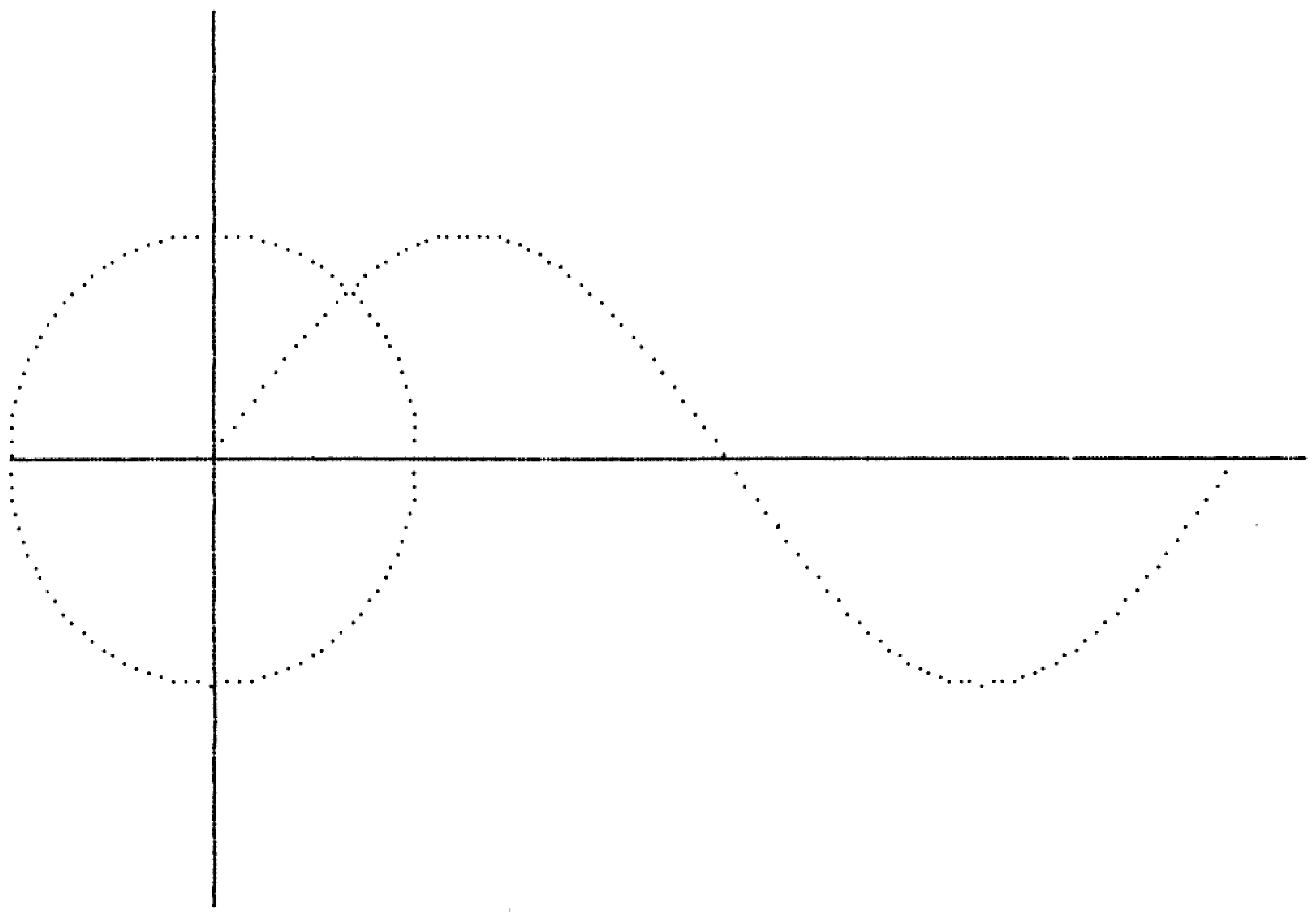
文　例　CMD PSET (30, 40), 1

解　説
- ビュー座標の(Hx, Hy)の位置にピクセルをセットします。
- カラーは〈カラーナンバ〉によって指定し，もし省略された場合は，現在のフォアグラウンドカラーナンバ (CMD COLOR参照) が用いられます。
- CMD PSET 実行後，LPは(Hx, Hy)に設定されます。

参　照　CMD COLOR, CMD PRESET

サンプルプログラム

```
100 'CMD PSET
110 CMD SCREEN 2,0
120 CMD CLS 2
130 CMD LINE(0,100)-(319,100),1
140 CMD LINE(50,0)-(50,200),1
150 'CIRCLE
160 P=3.14159
170 FOR T=0 TO 2*P STEP 2*P/100
180    X=50+50*COS(T):Y=100-50*SIN(T)
190   CMD PSET(X,Y)
200   X=50+40*T
210   CMD PSET (X,Y)
220 NEXT T
230 END
```



CMD PUT@ N80 グラフィックステートメント

(機能) **高解像度グラフィックパターンや漢字をグラフィック画面に表示します。**

(書式) **1) CMD PUT〔@〕(Hx, Hy), 〈配列変数名〉〔(〈要素ナンバ〉)〕〔, 〈機能〉〕〔, 〈フォアグラウンドカラーナンバ〉, 〈バックグラウンドカラーナンバ〉〕**

2) CMD PUT〔@〕(Hx, Hy), KANJI(〈漢字コード〉),〔,〈機能〉〕〔,〈フォアグラウンドカラーナンバ〉, 〈バックグラウンドカラーナンバ〉〕

(文例) **1) CMD PUT (100, 100), G%, PSET**

2) CMD PUT (0, 0), KANJI (&H3A34)

(解説) 1) • CMD GET@によって配列に読み込まれた高解像度グラフィックパターンをグラフィック画面上の任意の位置に表示します。

• 座標(Hx, Hy)はCMD GET@の場合と同様，ビュー座標で指定します。またこの命令を実行するとLPは(Hx, Hy)に移動されます。

• 〈配列変数名〉は表示したいグラフィックパターンが格納されている配列名であり，〈要素ナンバ〉は配列内のどこからデータを取り始めるかの指定です。〈要素ナンバ〉が省略された場合は、配列の最初から取り始めます。これらの指定は，CMD GET@命令で使ったものと同じものを指定します。

• 〈機能〉とは，グラフィックパターンを画面に表示する際，いろいろな機能を指定できるもので，次に示すものが用意されています。

PSET　　配列内のグラフィックパターンをそのまま表示します。

PRESET	白黒モードの場合は配列内のパターンをリバースして表示します。4色カラーモードの場合は各ピクセルのカラーナンバを，3－(そのピクセルのカラーナンバ)として表示します。
OR	配列内のグラフィックパターンと，既にある画面上のグラフィックパターンを1ビット(ピクセル)ごとにOR(論理和)し，その結果を画面に表示します。
AND	配列内のパターンと画面上のパターンをピクセルごとにAND(論理積)し，その結果を画面上に表示します。
XOR	配列内のパターンをピクセルごとにXOR(排他的論理和)し，その結果を画面に表示します。

※これらの〈機能〉は画面モードによって演算の対象が違います。モノクロモード，アトリビュートカラーモードの場合ピクセルがあるかないかを対象とし，カラーモードの場合カラーナンバを対象とします。もし〈機能〉が省略された場合はXORとみなされます。

- 最後の〈フォアグラウンドカラーナンバ〉，〈バックグラウンドカラーナンバ〉は，モノクロモード，アトリビュートカラーモードの時に読み込んだパターンをカラーモードにおいて表示する時にのみ有効なオプションパラメータです。この2つのパラメータは両方とも指定するか，両方とも指定しないかのどちらかしか許されません。
- 〈フォアグラウンドカラーナンバ〉は，モノクロモード・アトリビュートカラーモードで読み込んだ際にカラーナンバが

1に対しての色指定で，これを4色カラーモードで実行すると任意の色に変えることができます。〈バックグラウンドカラーナンバ〉は同様に黒であったピクセルに対しての色指定です。これらは共に0～3のカラーナンバによって指定します。

・CMD GET@とCMD PUT@命令は上記した〈フォアグラウンドカラーナンバ〉,〈バックグラウンドカラーナンバ〉を指定してモノクロモード，アトリビュートカラーモードで読み込んだパターンを4色カラーモードで表示する用途の他は，原則として同一画面モードで使用するようにしてください。

2)・　〈漢字コード〉で指定された漢字などの文字を画面に表示します。これらはオプションROMの中に格納されています。ユーザはこれらの中から任意の文字を〈漢字コード〉によって指定し，画面上に日本語の文章を作成することができます。

・　1)のCMD PUT@ではユーザがCMD GET@によって配列に読み込んだパターンを画面に表示しますが，漢字のCMD PUT@では配列内のデータにあたるものが，あらかじめ用意されています。このことを除けば，その使い方及び機能は1)の場合とほぼ同様です。ただし，漢字パターンは原則として縦16ピクセル，横16ピクセルの大きさと決められています(8×8，8×16のものもある。)また漢字パターンは白黒モードでCMD GET@した場合と同じ形で格納されていますから，4色カラーモード0/1において〈フォアグラウンドカラーナンバ〉，〈バックグラウンドカラーナンバ〉で指定してカラー表示させることも可能です。

・　1つのCMD PUT@では1つの漢字しか表示することができません。

参照　CMD GET@

サンプルプログラム

```
100 CMD SCREEN 2
110 CMD COLOR 2
120 AD=&H3020
130 CMD CLS 3
140 LOCATE 0,0:PRINT "JIS CODE =";HEX$(AD);"H"
150 FOR Y=22 TO 122 STEP 20
160   FOR X=2 TO 302 STEP 20
170     CMD PUT(X,Y),KANJI(AD)
180     AD=AD+1
190   NEXT
200 NEXT
210 CMD COPY 3
220 AD=AD+160
230 IF AD>&H735F THEN END
240 GOTO 130
250 END
```

```
JIS CODE =3020H

  亜唖娃阿哀愛挨姶逢葵茜穐悪握渥
旭葦芦鯵梓圧斡扱宛姐虻飴絢綾鮎或
粟袷安庵按暗案闇鞍杏以伊位依偉囲
夷委威尉惟意慰易椅為畏異移維緯胃
萎衣謂違遺医井亥域育郁磯一壱溢逸
稲茨芋鰯允印咽員因姻引飲淫胤蔭

cmd    auto    go to    list    run
```

CMD SCREEN N80 画面制御ステートメント

(機能) グラフィック画面に対して種々のモード設定をする。

(書式) CMD SCREEN〔〈グラフィックモード〉〕〔,〈グラフィックスイッチ〉〕〔,〈選択色のカラーコード〉〕

(文例) CMD SCREEN 2' 0' 3

(解説) • 〈グラフィックモード〉は，カラー，モノクロ，分解能と，グラフィック画面の最も基本的なモードを設定するもので次の値をとります。

0—モノクロモード(640×200ピクセル)

1—アトリビュートカラーモード(640×200ピクセル)

2—4色カラーモード 0 (320×200ピクセル)

3—4色カラーモード 1 (320×200ピクセル)

• 〈グラフィックスイッチ〉を0以外の数字にすると，現在グラフィック画面に描かれているパターンなどは一時的に消去されます。また0にすれば元に戻ります。

• 〈選択色のカラーコード〉は，カラーナンバと密接な関係にあり次のようになります。8色あるカラーコードの中から任意の1色をカラーナンバに対応させることができます。

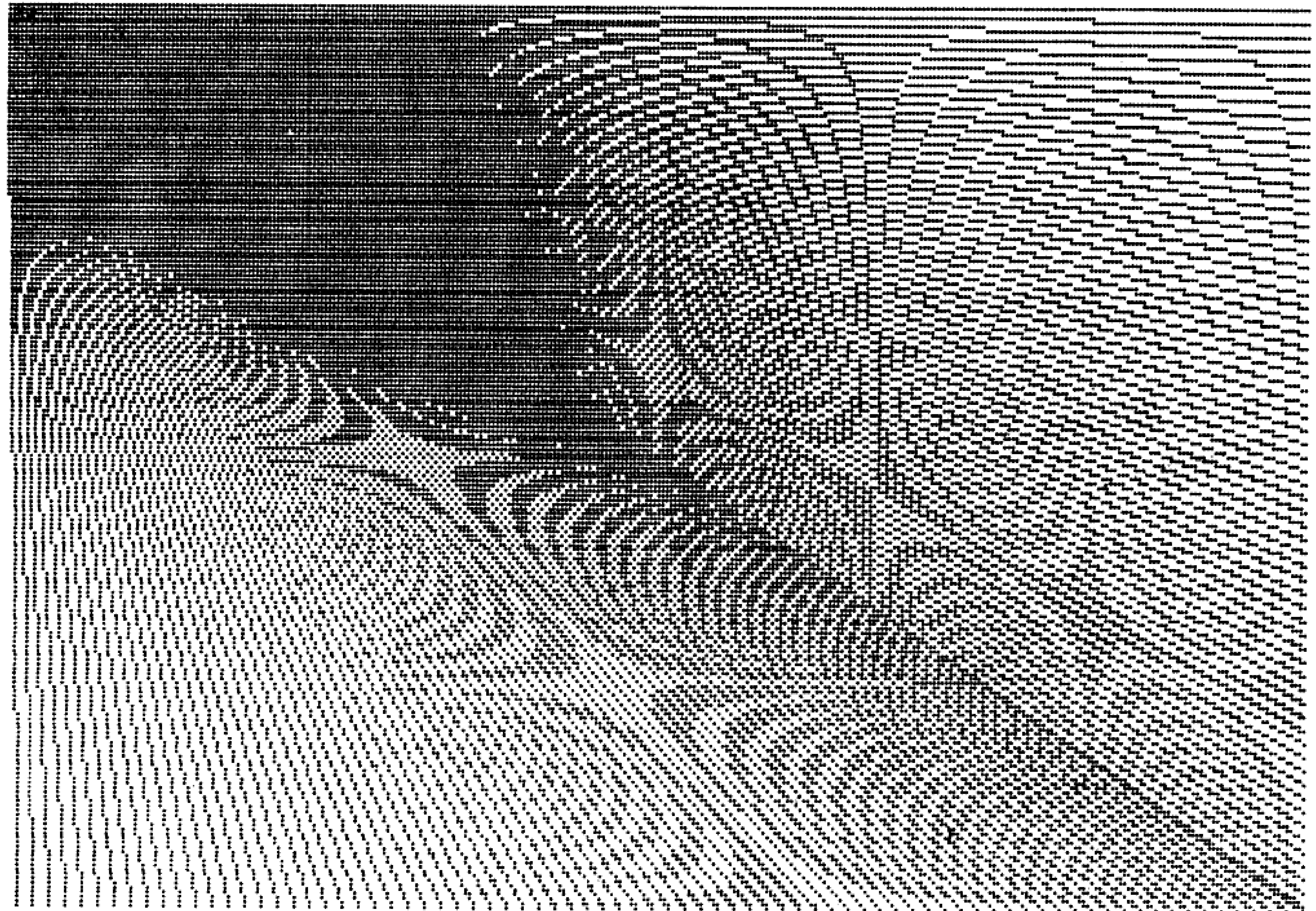
グラフィックモード / カラーナンバ	モノクロモード	アトリビュートカラーモード	4色カラーモード0	4色カラーモード1
0	黒	選択色(黒)	黒	青
1	選択色(白)	テキスト画面の色に依存します。	赤	マゼンタ
2	———	———	緑	シアン
3	———	———	選択色(青)	選択色(黒)

• CMD SCREENが実行されると，ビューポートはリセッ

トされ，画面のすべてが表示可能となります。

サンプルプログラム

```
100 CONSOLE 0,25,0,1
110 CMD SCREEN 0,,3
140 CMD COLOR 3,,0
150 CMD CLS 3
160 FOR I=0 TO 199 STEP 3
170   CMD LINE(0,0)-(639,I)
180 NEXT I
190 FOR I=639 TO 0 STEP -8
200   CMD LINE(0,0)-(I,199)
210 NEXT I
220 FOR I= 0 TO 3
225   CMD SCREEN I
230   FOR J=1 TO 800:NEXT J
240 NEXT I
260 GOTO 220
```



CMD VIEW  画面制御ステートメント

機能　グラフィック画面上での表示領域(ビューポート)を指定します。

書式　CMD VIEW (Hx1, Hy1)−(Hx2, Hy2)

文例　CMD VIEW(20, 20)−(300, 70)

解説

• (Hx1, Hy1)を左上の頂点(Hx2, Hy2)を右下の頂点とする絶対座標系上の長方形を高解像グラフィックス表示領域として指定します。この領域はビューポート(View Port)と呼ばれます。

• N80-BASICの種々の命令 (CMD PSET, CMD LINE, CMD CIRCLEなど)は，ビューポートの範囲外にグラフィックスを描くことができません。

• CMD VIEWは，グラフィックスの表示領域の指定を行うだけで，実際の画面に対する操作は行いません。したがって以前のビューポート中の図形が移動するようなことはありません。

• CMD VIEWの指定を変えるだけで，1つの図形を描くルーチンでも，画面上の異なる位置で描くようになります。

• Hx1<Hx2, Hy1<Hy2が成り立たない場合，あるいはこれらの座標が絶対座標の範囲から外れている場合には“Illegal function call”エラーが起こります。

• 一度設定されたビューポートは，次にCMD VIEWあるいは，CMD SCREENが実行されるまで変化しません。また，LPはビューポートの左上の頂点(Hx1, Hy1)に設定されます。

• リセットあるいはCMD SCREEN実行後のビューポートは次のように設定されています。

モノクロモード，アトリビュートカラーモードのとき
CMD VIEW (0,0)−(639,199)

4色カラーモード0，1のとき

CMD VIEW (0,0)−(319,199)

• CMD CLS 2, またはCMD CLS 3を実行するとグラフィック画面は，ビューポート内のみクリアされます。

参照　CMD CLS, CMD SCREEN, 1章・ビューポート

サンプルプログラム

```
10 CMD CLS 3
20 FOR I=0 TO RND(1)*20
30 FOR M=2 TO 3
40 FOR J=0 TO RND(1)*20
50 CMD SCREEN M,,RND(1)*8
60 CMD COLOR,RND(1)*4
70 CMD VIEW(RND(1)*320,RND(1)*200)-(RND(1)*320,RND(1)*200)
80 CMD CLS 2
90 CMD CIRCLE(10,10),10
100 NEXT
110 NEXT
120 NEXT
130 CMD SCREEN 2
140 CMD COLOR ,0
150 CMD CLS 2
160 END
```

COLOR　N80 N　画面制御ステートメント

機　能　**テキスト画面の色や機能を指定します。**

書　式　**COLOR〔〈ファンクションコード〉〕〔,〈ヌルキャラクタコード〉〕〔,〈グラフィックスイッチ〉〕**

文　例　**COLOR 3, 0, 1**

解　説

- テキスト画面に対するすべての操作に使われる色や機能を設定します。
- N80(N)-BASICではテキスト画面と低解像度グラフィック画面は同一の画面を使用しています。したがって，この命令によりキャラクタ及び低解像度グラフィックの機能がキャラクタ単位で設定されます。
- 〈ファンクションコード〉は，テキスト画面の文字にいろいろな機能を与えます。このファンクションコードは，現在テキスト画面がカラーモードになっているか，白黒モードになっているかによって働きが異なります。

白黒モードの場合 (CONSOLE,,,0)

0—ノーマル(通常の表示)
1—シークレット(文字は表示されない)
2—ブリンク(点滅する)
3—シークレット(1と同じ)
4—リバース(反転する)
5—リバースシークレット(反転して文字は表示されない)
6—リバースブリンク(反転して点滅する)
7—リバースシークレット(5と同じ)

カラーモードの場合(CONSOLE,,,1)

0—黒	1—青	2—赤	3—マゼンタ
4—緑	5—シアン	6—黄色	7—白

- 〈ヌルキャラクタコード〉はテキスト画面をクリアした場

合に表示されるキャラクタコードを指定します。通常は0を指定します。

• 〈グラフィックスイッチ〉を1にすれば低解像度グラフィックモードに，0にすればキャラクタモードになります。キャラクタモードで低解像度グラフィック命令を実行するとキャラクタ画面が乱れることがあります。したがって，低解像度グラフィックを使用するときは〈グラフィックスイッチ〉を1にしてください。

注　意　1行中で，20回以上違う色や機能を指定しても無視されます。また，キャラクタと低解像度グラフィックの混在も20カ所以上，変更点がある場合，動作が保障されません。

参　照　CONSOLE

サンプルプログラム

```
console ,,,0
Ok
color 4
Ok
print "N80-BASIC"
N80-BASIC
Ok

console ,,,1
Ok
color 4
Ok
print "N80-BASIC"
N80-BASIC
Ok
```

CONSOLE　N80 N　画面制御ステートメント

機能　テキスト画面のモード設定を行います。

書式　CONSOLE〔〈スクロール開始行〉〕，〔〈スクロール行数〉〕〔，〈ファンクションキー表示スイッチ〉〕〔，〈カラー/白黒スイッチ〉〕

文例　CONSOLE 0, 25, 0, 1

解説

- 〈スクロール開始行〉と〈スクロール行数〉を指定することにより，画面上でスクロールする領域(スクロールウィンドウ)を指定します。
- 〈スクロール開始行〉は，どの行からスクロールを開始するかを指定します。なお画面の最上行が0となります。
- 〈ファンクションキー表示スイッチ〉に1を指定すると画面最下行に，ファンクションキーに登録されている文字列を表示します。0を指定した場合，この表示はされません。
- 〈カラー／白黒スイッチ〉を1に指定するとテキスト画面をカラーモードにし，0を指定した場合には白黒モードとなります。
- リセット後の初期化された状態では，〈スクロール開始行〉，〈スクロール行数〉，〈ファンクションキー表示スイッチ〉，〈カラー／白黒スイッチ〉はそれぞれ0，20，1，0に設定されます。また，それぞれのパラメータを省略したときは，現在定義されている値と同じになります。

サンプルプログラム

```
110 PRINT CHR$(12):'CMD CLS 1
120 CONSOLE 0,25,0,1
130 FOR I=1 TO 25 STEP 4
140   CONSOLE 0,I
150   GOSUB 190
160 NEXT I
170 END
180 '-- printsub --
190 PRINT CHR$(12):'cmd cls
200 FOR J=1 TO 40
210   PRINT J
220   FOR K=0 TO 10:NEXT K
230 NEXT J
240 RETURN
```

CONT　N80 N　一般コマンド

機能　**ストップキー入力後，または STOP によって停止したプログラムの実行を再開します。**

書式　**CONT**

文例　**CONT**

解説

- CONT は通常プログラムの誤りを直すために STOP と共に用います。
- プログラムの実行停止後，ダイレクトモードで CONT RETURN を入力すると停止した次の文から実行が再開されます。ただし，INPUT, LINE INPUT で実行が停止した場合，それらの命令から実行が再開されます。
- 実行を停止後 ダイレクトモードで変数の値を変更したり，表示した後，CONT によって実行を再開することができます。ただし，プログラムの変更があったときはエラーとなります。
- STOP の他に，END で終了したプログラムも CONT で実行を再開することができます。

サンプルプログラム

```
100 'CONT test
110 I=10
120 PRINT I,SQR(I)
130 STOP
140 I=I+1
150 GOTO 120

run
 10            3.16228
Break in 130
Ok
cont
 11            3.31663
Break in 130
Ok
cont
 12            3.4641
Break in 130
Ok
cont
 13            3.60555
Break in 130
Ok
```

COS　N80 N　数値関数

機能　**余弦(コサイン)の値を与えます。**

書式　**COS**(〈数式〉)

文例　**A＝COS(C/B)**

解説

- 〈数式〉の単位をラジアン(π/180×角度)としたときの余弦(コサイン)を与えます。
- COS 関数の演算は単精度で行われます。

サンプルプログラム

```
100 '-- COS ｶﾗ SEC(ｾｶﾝﾄ) ｦ ﾓﾄﾒﾙ --
110 INPUT "ｶｸﾄﾞ (ﾄﾞ) : ";D
120 CS=COS(D/180*3.14159265358979#)
130 PRINT "COS";USING "###";D;:PRINT " = ";CS,
140 PRINT "SEC";USING "###";D;
150 IF ABS(CS)<=1E-06 THEN 180 ELSE SC=1/CS
160 PRINT " = ";SC
170 END

run
ｶｸﾄﾞ (ﾄﾞ) : ? 45
COS 45 =  .707107           SEC 45 =  1.41421
Ok
```

CSAVE　N80 N　入出力コマンド

機能　メモリからカセットテープへプログラムをセーブします。

書式　CSAVE<ファイル名>

文例　CSAVE "DEMO"

解説
- CSAVE はメモリ上のプログラムを指定された<ファイル名>でカセットへセーブする命令です。
- <ファイル名>は 6 文字以内で指定してください。
- プログラムのセーブが完了すると，画面に"Ok"と表示してコマンドレベルに戻ります。

サンプルプログラム

```
csave"sumo"
Ok
```

CSNG　N80 N　数値関数

機能　**整数値，倍精度実数値を単精度実数値に変換します。**

書式　**CSNG(〈数式〉)**

文例　**A!=CSNG(C#/2)**

解説

- 〈数式〉の値を有効数字の6桁の単精度実数値に変換します。
- 結果の値が－1.70141E＋38～1.70141E＋38の範囲にないときには，“Overflow”エラーが起ります。

参照　CINT, CDBL

サンプルプログラム

```
100 '-- ｼｮｳｽｳﾃﾝｲｶ ﾆｼﾞｲ ﾉ ｹﾀ ﾃﾞﾉ 4ｼｬ5ﾆｭｳ --
110 A#=1.23456789#:D=1:GOSUB 140:GOSUB 170
120 A#=1.23456789#:D=7:GOSUB 140:GOSUB 170
130 END
140 IF A#=INT(A#) THEN AD#=A#:RETURN
150 AD#=INT(A#*10^D+.5)/10^D
160 RETURN
170 PRINT A#,CSNG(AD#)
180 RETURN

run
 1.23456789     1.2
 1.23456789     1.23457
Ok
```

CSRLIN　N80 N　画面制御予約変数

機能　**テキスト画面上の現在のカーソルの垂直位置を与えます。**

書式　**CSRLIN**

文例　**Y=CSRLIN**

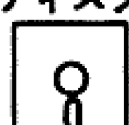
解説

- 現在のカーソルの垂直位置を行単位で与えます。
- 得られる値は，25行モードで０から24，20行モードでは０から19となります。なお，画面の最上行が０，最下行が24(または19)となっています。

参照　POS

サンプルプログラム

```
100 '-- ﾋﾞﾘﾔｰﾄﾞ ﾉ ｼﾐｭﾚｰｼｮﾝ --
101 WIDTH 80,25:CMD CLS 1
110 LOCATE 1,2:PRINT STRING$(78,"●");
120 LOCATE 1,23:PRINT STRING$(78,"●");
130 FOR I=3 TO 22:LOCATE 0,I:PRINT "●";
140 LOCATE 79,I:PRINT "●";:NEXT
150 LOCATE 0,2:PRINT "●";:LOCATE 79,2:PRINT "●";
160 LOCATE 0,23:PRINT "●";:LOCATE 79,23:PRINT "●";
170 F=1:X=RND(1)*77+1:C=RND(1)*18+3
180 FOR X=X TO 78
190   LOCATE X0,C0:PRINT " ";
200   IF CSRLIN=22 OR CSRLIN=3 THEN F=F*-1
210   C=C+F:LOCATE X,C:PRINT "●";:X0=X:C0=C
220 NEXT X
230 FOR X=78 TO 2 STEP -1
240   LOCATE X0,C0:PRINT " ";
250   IF CSRLIN=22 OR CSRLIN=3 THEN F=F*-1
260   C=C+F:LOCATE X,C:PRINT "●";:X0=X:C0=C
270 NEXT X
280 GOTO 180
```

CVI/CVS/CVD ディスク    数値関数

機能 文字列で表された値を数値データに変換します。

書式
1) **CVI(〈2文字の文字列〉)**
2) **CVS(〈4文字の文字列〉)**
3) **CVD(〈8文字の文字列〉)**

文例
1) **A%=CVI (A$)**
2) **B=CVS("A3BD")**
3) **C#=CVD(NM$)**

解説
• フロッピィディスク上のランダムアクセスファイルから読み込んだデータは文字型になっているため CVI, CVS, CVD 関数を使って，数値データに変換しなければなりません。

• CVI は 2 文字の文字列(即ち 2 バイトのデータ)を整数値に，CVS は 4 文字の文字列を単精度実数値に，CVD は 8 文字の文字列を倍精度実数値にそれぞれ変換します。

参照 MKI$/MKS$/MKD$

サンプルプログラム

```
100 '-- input value from random file --
110 OPEN "cvsmks" AS #1
120 FIELD #1,2 AS A1$,4 AS A2$,8 AS A3$
130   GET #1,1:A1%=CVI(A1$):A2!=CVS(A2$):A3#=CVD(A3$)
140   PRINT A1%,A2!,A3#
150 END

run
 32767          1.23456          3.14159265358979
Ok
```

DATA N80 N 一般ステートメント

機能 **READで読み込まれる数値，文字定数を格納します。**

書式 **DATA 〈定数〉〔, 〈定数〉…〕**

文例 **DATA 1, NEC, PC-8001MK2**

解説

- DATAは非実行文で，プログラムのどこにおいてもかまいません。
- 1つのDATAには，1行(255文字)に入るだけのデータをセットすることができ，また1つのプログラムの中にいくつものDATAを置くことができます。
- READは，行番号の小さい方から順番にDATAの中のデータを読み込んでいきます。
- READされる回数よりデータの個数が少ない場合(READに対応するデータがない場合) "Out of data" エラーが起ります。
- 〈定数〉は，数値定数(整数，固定小数点，浮動小数点，16進数，8進数)，文字定数のいずれかです。ただし，定数式(2 * 3など)は許されません。また，READで読み込まれる場合，READで指定されている変数の型は，対応するデータの定数の型と一致しなければなりません。
- DATAの中のデータは，コンマ(,)で区切られますが，文字定数の中にコンマ，コロン(:)，または文字列の前後に意味のある空白を含むときは，その文字定数全体をダブルクォーテーションマーク(")で囲む必要があります。
- RESTOREによってREADで読み込むDATAを行単位で指定することができます。

サンプルプログラム

```
100 DATA 10,10,   PC-8001MK][,"   PC-8001MK]["
110 READ A
120 READ B$:B=VAL("&H"+B$)
130 READ C$:READ D$
140 PRINT A:PRINT B
150 PRINT C$:PRINT D$

run
 10
 16
PC-8001MK][
   PC-8001MK][
Ok
```

DATE$ [N80] [N] 文字予約変数

機能 内蔵のカレンダ時計の日付を与えます。

書式 DATE$

文例 PRINT DATE$

解説 • DATE$には内蔵カレンダ時計の日付が入れられており，文字変数に代入したり，PRINTで内容を表示することができます。

• ユーザーはDATE$に日付をセットすることもできます。文字定数の型式はYY/MM/DD (例："83/01/01")となり，それぞれ2文字で年，月，日をそれぞれ指定します。

• PC-8001MKII内蔵のカレンダ時計はバッテリーバックアップにより常に日付を更新しています。

サンプルプログラム

```
100 '-- example  "83/12/25" --> "25,Dec.1983" --
110 DIM MN$(12)
120 FOR I=1 TO 12
130   READ MN$(I)
140 NEXT I
150 INPUT "date (YY/MM/DD) ";D$
160 DATE$=D$
170 GOSUB 210
180 PRINT DATE$;" --> ";M$
190 END
200 'trns
210 M$=RIGHT$(DATE$,2)+","
220 M$=M$+MN$(VAL(MID$(DATE$,4,2)))
230 M$=M$+".19"+LEFT$(DATE$,2)
240 RETURN
250 DATA Jan,Feb,Mar,Abr,May,Jun
260 DATA Jun,Aug,Sep,Oct,Nov,Dec

run
date (YY/MM/DD) ? 83/01/01
83/01/01 --> 01,Jan.1983
Ok
```

DEF FN　[N88] [N]　一般ステートメント

機能　ユーザによって作られた関数を定義します。

書式　DEF FN〈名前〉〔(〈パラメータリスト〉)〕=〈関数の定義式〉

文例　DEF FNA(X, Y)=SQR(X^2+Y^2)

解説

- 関数の名前は“FN”+〈名前〉で定義されます。
- 〈名前〉は変数名として正しい形をしたものでなければなりません。変数名は最初の2文字までの判別しかしません。
- 〈パラメータリスト〉はその関数が呼ばれたときに，〈関数の定義式〉中の同じ変数名に対応し，変数名の間はコンマで区切ります。
- 〈パラメータリスト〉に書かれた変数名は〈関数の定義式〉を評価する際にのみ有効となります。したがってプログラム中に同じ変数名があっても影響はありません。

サンプルプログラム

```
100 DEF FNFU(G,R,N)=G*(1+R)^N
110 DEF FNTA(G,R,N)=G*(1+R*N)
120 INPUT "ｶﾞﾝｷﾝ  :";GA:INPUT "ﾈﾝﾘ     :";NN:INPUT "ﾈﾝｽｳ    :";NE
140 FG=FNFU(GA,NN,NE)
150 TG=FNTA(GA,NN,NE)
160 F$="¥¥#########,. "
170 PRINT "ﾌｸﾘ ﾃﾞﾊ ";USING F$;FG
180 PRINT "ﾀﾝﾘ ﾃﾞﾊ ";USING F$;TG

run
ｶﾞﾝｷﾝ  :? 100000
ﾈﾝﾘ    :? .07
ﾈﾝｽｳ   :? 10
ﾌｸﾘ ﾃﾞﾊ     ¥196,715.
ﾀﾝﾘ ﾃﾞﾊ     ¥170,000.
Ok
```

DEF INT/SNG/DBL/STR　N80 N

一般ステートメント

機　能　**変数の型宣言をします。**

書　式

```
DEF | INT | <文字の範囲> [, <文字の範囲>……]
    | SNG |
    | DBL |
    | STR |
```

文　例　**DEFINT A, I—N**

解　説

• 〈文字の範囲〉で指定された文字で始まるすべての変数名の型を，次に示す型宣言により定義します。

DEFINT：変数を整数型に
DEFSNG：変数を単精度型に
DEFDBL：変数を倍精度型に
DEFSTR：変数を文字型に
｝宣言します。

• 〈文字の範囲〉は，1文字の英字で指定します。またある範囲にわたって指定するときは，1文字の英字と1文字の英字をハイフン(－)でつなぎます。但し，最初の英字は，後の英字よりもアルファベット順で先のものでなければなりません。

• 型宣言文字による指定がある場合，これらのDEF型宣言よりも優先されます。

• 型宣言が行われていない文字で始まる変数は，すべて単精度変数とみなされます。

サンプルプログラム

```
100 DEFINT I-M
110 DEFSNG A,B
120 DEFDBL D
130 I1=1.23:J=65.643
140 AB=1.23:BB=65.643
150 D=3.141592653589792#
160 D%=3.141592654000001#
170 PRINT "I1 =";I1,,"j=";J
180 PRINT "AB =";AB,,"BB =";BB
190 PRINT "D =";D,"D% =";D%
200 END
```

```
run
I1 = 1                          j= 65
AB = 1.23                       BB = 65.643
D = 3.141592653589792           D% = 3
Ok
```

DEF USR　[N80] [N]　特殊ステートメント

機能　**機械語で作られたユーザ関数の実行開始番地を定義します。**

書式　**DEF USR〔<番号>〕=<開始番地>**

文例　**DEF USR3=&HDA00**

解説

• USR関数(ユーザ関数)が呼び出す，機械語ルーチンの実行開始番地の設定を行います。

• <番号>は0～9までの値で，USR+<番号>が関数名となり，関数名によって機械語で作られたプログラムが呼び出されます。<番号>を省略したときは0とみなされます。

• <開始番地>は機械語プログラムの実行開始番地を指定します。また，ユーザが作成した機械語プログラムを呼び出す場合，<開始番地>は，CLEARの<メモリの上限>で指定した番地より後でなければなりません。

• USR関数の作り方については，付録B 機械語プログラムの作り方を参照してください。

参照　USR，付録B 機械語プログラムの作り方

サンプルプログラム

```
100 CLEAR 300,&HBFFF
110 DEF USR=&HC000
120 A=USR(1)
130 END
```

DELETE　N80 N　一般コマンド

機能　**プログラムの行を削除します。**

書式　**DELETE〈始点行番号〉〔−〈終点行番号〉〕**

文例　**DELETE 500−620**

解説

- 〈始点行番号〉から〈終点行番号〉までのプログラムを削除します。
- 〈始点行番号〉だけを指定した場合はその行だけを削除します。この場合 DELETE を省略して行番号だけの入力と同じになります。
- ハイフンと〈終点行番号〉のみを指定した場合，プログラムの先頭から指定行までを削除します。

サンプルプログラム

```
100 ' DELETE ｺﾏﾝﾄﾞ ﾉ ｻﾝﾌﾟﾙ ﾌﾟﾛｸﾞﾗﾑ
110 '
120 ' DELETE ｺﾏﾝﾄﾞ ﾊ ｼﾃｲ ｼﾀ ｷﾞｮｳ ｦ
130 ' ｻｸｼﾞｮ ｼﾏｽ
140 PRINT "DELETE ｺﾏﾝﾄﾞ ﾉ ｻﾝﾌﾟﾙ ﾌﾟﾛｸﾞﾗﾑ"
150 PRINT "DELETE ｺﾏﾝﾄﾞ ﾊ ｼﾃｲ ｼﾀ ｷﾞｮｳ ｦ"
160 PRINT "ｻｸｼﾞｮ ｼﾏｽ"
170 END

Ok
delete 110-130
Ok
list
100 ' DELETE ｺﾏﾝﾄﾞ ﾉ ｻﾝﾌﾟﾙ ﾌﾟﾛｸﾞﾗﾑ
140 PRINT "DELETE ｺﾏﾝﾄﾞ ﾉ ｻﾝﾌﾟﾙ ﾌﾟﾛｸﾞﾗﾑ"
150 PRINT "DELETE ｺﾏﾝﾄﾞ ﾊ ｼﾃｲ ｼﾀ ｷﾞｮｳ ｦ"
160 PRINT "ｻｸｼﾞｮ ｼﾏｽ"
170 END
Ok
```

DIM　[N80] [N]　一般ステートメント

機能　**配列変数の要素の大きさを指定し，メモリ領域に割り当てます。**

書式　**DIM〈変数名〉(〈添字の最大値〉〔,〈添字の最大値〉…〕)**

文例　**DIM A(10, 10)**

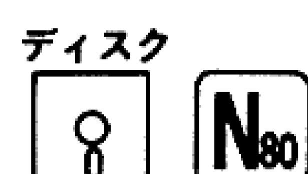
解説

- 配列変数の添字の最大値を設定し，同時にメモリ上にその配列の領域を割り当てます。
- 〈添字の最大値〉の個数は，その配列変数の次元を示します。また,次元はメモリの許す限り255次元まで使用できます。
- DIM を宣言せずに配列変数を用いた場合，その添字の最大値は10とみなされます。
- 〈添字の最大値〉より大きい値の添字が用いられた場合は，“Subscript out of range” のエラーが起こります。
- また，DIM が実行された時点で，その配列のすべての要素の値は 0 (文字型の場合はヌルストリング)に設定されます。

サンプルプログラム

```
100 '    ｸｸ ﾉ ﾋｮｳ ｦ ﾂｸﾙ
110 DIM KU(9,9)
130 FOR I=1 TO 9:FOR J=1 TO 9
140   KU(I,J)=I*J
150 NEXT J,I
160 FOR I=1 TO 9:FOR J=1 TO 9
170   PRINT USING "####";KU(J,I);
180 NEXT J:PRINT
190 NEXT I
200 END

run
   1   2   3   4   5   6   7   8   9
   2   4   6   8  10  12  14  16  18
   3   6   9  12  15  18  21  24  27
   4   8  12  16  20  24  28  32  36
   5  10  15  20  25  30  35  40  45
   6  12  18  24  30  36  42  48  54
   7  14  21  28  35  42  49  56  63
   8  16  24  32  40  48  56  64  72
   9  18  27  36  45  54  63  72  81
Ok
```

(1)DSKF  入出力関数

(機能) **フロッピィディスクの情報を与えます。**

(書式) **DSKF(〈ドライブ番号〉〔,〈機能〉〕)**

(文例) **PRINT DSKF(1)**

(解説) • 〈ドライブ番号〉で指定されたドライブに入っているフロッピィディスクに関する情報を関数の値として返します。

• 〈機能〉は値として返す情報の種類を指定します。〈機能〉として指定する値と，その時に返される関数の値の意味は次のようになっています。

省略された時：フロッピィディスクの残り容量をクラスタ単位で返す。

0：…………最大トラック番号（＝片面当りのトラック数－1）

1：…………1トラック当りのセクタ数

2：…………片面フロッピィディスク(0を返す)あるいは両面フロッピィディスク(1を返す)

3：…………1トラック当りのクラスタ数

4：…………ボリューム当りのクラスタ数

5：…………ディレクトリトラック番号

6：…………1クラスタ当りのセクタ数

7：…………FATの開始セクタ番号

8：…………FATの終了セクタ番号

9：…………FATの数

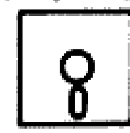
10：…………IDのあるセクタ番号

• ドライブの種類やディレクトリ，クラスタ，FATについては，PC－8001MKⅡユーザーズマニュアルを参照してください。

サンプルプログラム

```
100 INPUT "drive number";DR
110 PRINT "Track  =";DSKF(DR,0);"/Drive"
120 PRINT "Sector =";DSKF(DR,1);"/Track"
130 PRINT "Disk space =";DSKF(DR)

run
Track  = 34 /Drive
Sector = 19 /Track
Disk space = 26
Ok
```

(2)DSKF　 　入出力関数

(機　能) **フロッピィディスクの残り容量を与えます。**

(書　式) **DSKF(〈ドライブ番号〉)**

(文　例) **PRINT DSKF(1)**

(解　説) • 〈ドライブ番号〉で指定されたフロッピィディスクの残り容量をクラスタ単位で返します。

• DISK－BASICではミニフロッピィディスクだけしか使用できませんから1クラスタ＝8セクタ(2Kバイト)に固定されています。

(参　照) (1)DSKF

(サンプルプログラム)

```
print dskf(1)
 45
Ok
print dskf(2)
 59
Ok
```

DSKI$  入出力関数

機能 **フロッピィディスクから直接データを読み込みます。**

書式 1) **DSKI$(〈ドライブ番号〉, 〈トラック番号〉, 〈セクタ番号〉)**

2) **DSKI$(〈ドライブ番号〉, 〈ヘッド番号〉, 〈トラック番号〉, 〈セクタ番号〉)**

文例 1) **D$=DSKI$(1, 18, 3)**

2) **D$=DSKI$(2, 0, 19, 1)**

解説 • 通常のファイル操作 (OPEN, CLOSE) とは無関係に，フロッピィディスク上の指定したセクタから直接データを読み込みます。

• DSKI$ は，指定されたセクタ上に書かれているデータ256バイトを0番バッファに読み込み，最初の256文字の文字列を関数の値として返します。

• 文字列の長さは255文字までしか許されませんから，DSKI$関数の値として256バイトのデータすべてを1度に得ることはできません。そこで，ランダムアクセスの場合と同じように FIELD により0番バッファへ複数の文字変数を割り当てることにより解決できます。または，VARPTR 関数により，0番バッファの置かれているメモリ番地を調べた上で PEEK 関数によりバッファからデータを読み出すこともできます。

• 〈ヘッド番号〉は，指定された〈ドライブ番号〉のフロッピィディスクが両面の場合にのみ必要で，0または1で指定します。

• 〈トラック番号〉, 〈セクタ番号〉として指定できる値の範囲は，指定された〈ドライブ番号〉のディスクドライブの種類によって異なりますが，BASIC は，これらの値の範囲を自動

的に調べ，不当であった場合には“Illegal function call” または“Bad surface/track/sector”エラーが起ります。

- DSKI$ は対象とするディスクドライブが片面ディスクであるか両面ディスクであるかによって書式が変ります。

  1)の書式は片面フロッピィディスクを対象とする場合であり，〈ヘッド番号〉の指定は必要ありません。

  2)の書式は両面フロッピィディスクを対象とする場合であり，〈ヘッド番号〉を指定しなければなりません。

注　意　N80 DISK－BASIC 使用時に複数のディスクが接続されている場合，それぞれのディスクの情報についてはDSKF により調べることができます。

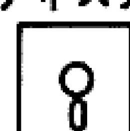
参　照　DSKO$ FIELD, (1)DSKF, VARPTR, PC-8001MKII ユーザーズマニュアル

サンプルプログラム

```
100 '-- read from disk (driect)--
110 FIELD #0,128 AS A$(0),128 AS A$(1)
120 INPUT "Drive";DR
130 IF DR<1 OR DR>8 THEN 120
140 IF DSKF(DR,2)=0 THEN 170:INPUT "Track,Sector";TR,SE
150 INPUT "Surface";SU
160 IF SU<0 OR SU>1 THEN 150
170 INPUT "Track,Sector";TR,SE
180 IF TR<0 OR TR>DSKF(DR,0) THEN 170
190 IF SE<1 OR SE>DSKF(DR,1) THEN 170
200 IF DSKF(DR,2)=0 THEN 220
210 DU$=DSKI$(DR,SU,TR,SE):GOTO 230
220 DU$=DSKI$(DR,TR,SE)
230 FOR I=0 TO 1
240   FOR J=0 TO 7
250     FOR K=1 TO 16
260       D$=HEX$(ASC(MID$(A$(I),J*16+K,1)))
270       PRINT RIGHT$("0"+D$,2);" ";
280     NEXT K:PRINT
290 NEXT J,I:END

Drive? 1
Surface? 0
Track,Sector? 18,1
09 00 0A 00 3A 8F E9 00 48 00 14 00 3A 8F E9 20
20 20 4E 45 43 20 50 43 2D 38 38 30 31 20 20 20
44 65 6D 6F 6E 73 74 72 61 74 69 6F 6E 20 70 72
6F 67 72 61 6D 20 20 28 32 29 20 20 20 22 44 45
4D 4F 55 54 59 22 00 50 00 1E 00 3A 8F E9 00 7B
00 28 00 9E 20 0F 28 2C 0F 14 3A 9D 20 11 2C 0F
14 2C 11 2C 12 3A CB 20 18 3A D3 20 11 2C 14 3A
```

```
CE 20 14 3A D3 20 11 2C 11 00 8B 00 32 00 8D 20
F5 53 55 42 43 4F 4C 4F 52 00 BF 00 3C 00 D6 20
11 2C 11 3A 91 20 22 4E 45 43 20 50 43 2D 38 38
30 31 20 44 65 6D 6F 6E 73 74 72 61 74 69 6F 6E
20 70 72 6F 67 72 61 6D 20 28 32 29 22 00 F3 00
46 00 D6 20 11 2C 14 3A 91 20 22 47 72 61 70 68
69 63 73 20 77 6F 72 6C 64 20 6F 66 20 36 34 30
2A 32 30 30 20 61 6E 64 20 36 34 30 2A 34 30 30
22 00 0D 01 50 00 D6 20 11 2C 16 3A 91 20 22 72
Ok
```

DSKO$    入出力ステートメント

(機　能) **フロッピィディスクに，直接データを書き込みます。**

(書　式) **1) DSKO$<ドライブ番号>,<トラック番号>,<セクタ番号>**

2) DSKO$<ドライブ番号>, <ヘッド番号>, <トラック番号>,<セクタ番号>

(文　例) **1) DSKO$ 2, 10, 1**

2) DSKO$ 1, 0, 19, 1

(解　説) • 通常のファイル操作 (OPEN, CLOSE) とは無関係に，フロッピィディスク上の指定したセクタへ直接データを書き込みます。

• DSKO$ は既存のフロッピィディスクファイルを壊す恐れがあります。したがってフロッピィディスク及びファイルの構成を正しく理解した上で使用してください。これらについては PC-8001 MKⅡユーザーズマニュアルを参照してください。

• DSKO$は，そのパラメータにより指定されたセクタに，0番バッファの内容256バイトを書き込みます。0番バッファは，通常のディスクファイル操作するときには用いることができませんが，DSKI$と DSKO$による直接操作のときだけ，ユーザが用いることができます。

• 書き込むデータの準備は，ランダムアクセスの場合と同じように FIELD により 0 番バッファへ変数領域を割り当てた後，LSET, RSET により行います。または，VARPTR 関数により，0 番バッファの置かれているメモリ番地を調べた上で，POKE によりバッファ中にデータを準備することもできます。

• <ヘッド番号>は，指定された<ドライブ番号>のフロッピィ

ディスクが両面用の場合にのみ必要で，0または1で指定します。

• 〈トラック番号〉，〈セクタ番号〉として指定できる値の範囲は，指定された〈ドライブ番号〉のフロッピィディスクドライブの種類によって異なりますが，BASICは，これらの値の範囲を自動的に調べ，不当であった場合には“Illegal function call”または“Bad surface/track/sector”エラーが起ります。

• DSKO$は対象とするディスクドライブが片面用であるか両面用であるかによって書式が変ります。

1)の書式は片面フロッピィディスクを対象とする場合であり，〈ヘッド番号〉の指定は必要ありません。

2)の書式は両面フロッピィディスクを対象とする場合であり，〈ヘッド番号〉を指定しなければなりません。

注　意　N_{80} DISK-BASIC使用時に，複数のディスクが接続されている場合，それぞれのディスクの情報についてはDSKFにより調べることができます。

参　照　DSKI$, FIELD, (1)DSKF, VARPTR, PC-8001MKIIユーザーズマニュアル

サンプルプログラム

```
100 '-----          Write to the disk          -----
110 '**  ｺﾉ program ﾊ disk ﾆ ﾃﾞｰﾀ ｦ ﾀﾞｲﾚｸﾄ ﾆ **
120 '**  ｶｷｺﾐ ﾏｽ ｶﾗ ﾌﾖｳｲ ﾆ ｼﾖｳ ｼﾅｲﾃﾞ ｸﾀﾞｻｲ    **
130 FIELD #0,128 AS A$,128 AS B$
140 LSET A$=STRING$(128,CHR$(&HFF))
150 LSET B$=STRING$(128,CHR$(&HFF))
160 'strat
170 INPUT "Drive";DR
180 IF DR<1 OR DR>8 THEN 170
190 IF DSKF(DR,2)=0 THEN 230
200 INPUT "Surface";SU
210 IF SU<0 OR SU>1 THEN 200
220 'reent
230 INPUT "Track,Sector";TR,SE
240 IF TR<0 OR TR>DSKF(DR,0) THEN 230
250 IF SE<1 OR SE>DSKF(DR,1) THEN 230
260 IF DSKF(DR,2)=0 THEN 290
270 DSKO$ DR,SU,TR,SE:GOTO 300
280 'single
290 DSKO$ DR,TR,SE
300 END
```

END　N80 N　一般ステートメント

機能　**プログラムの終了を宣言します。**

書式　**END**

文例　**END**

解説

• プログラムの実行を終了し，すべてのファイルをクローズしてコマンドレベルに戻ります。

• ENDはプログラムの実行を終了させるために，プログラム中のどこに置いてもかまいません。

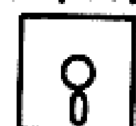
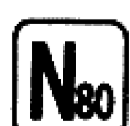
• プログラムの最後の END は省略することができます。ただし，この場合ファイルのクローズは行われません。

注意　カセットテープにセーブするプログラムは最後に END をつける習慣をつけてください。END がないとロードしたときに実行できないことがあります。

サンプルプログラム

```
100 WIDTH 80,25
110 COLOR 7
130 '-- loop --
140 PRINT  TIME$
150 A$=INKEY$
160 IF A$="s" THEN END
170 GOTO 140

run
13:53:20
13:53:20
13:53:21
13:53:21
13:53:21
Ok
```

EOF ディスク    入出力関数

機能 ファイルが終了したかどうかを検出します。

書式 EOF(〈ファイル番号〉)

文例 IF EOF(1) THEN CLOSE #1 : END

解説 ・シーケンシャルファイルにおいて〈ファイル番号〉で指定されたファイルが終りに達したかどうかを調べる関数です。

・ファイルが終わりに達したとき，EOF 関数は真(－1)，そうでなければ偽(0)を返します。

サンプルプログラム

```
100 '-- data input --
110 OPEN "2:data5" FOR INPUT AS #1
120 IF EOF(1) THEN 160
130   INPUT #1,A
140   PRINT A
150 GOTO 120
160 CLOSE

run
 1
 2
 3
 4
 5
 6
 7
 8
 9
 10
Ok
```

ERASE　N80 N　一般ステートメント

機能　配列変数を消去します。

書式　ERASE〈配列名〉〔,〈配列名〉……〕

文例　ERASE D, DA$

解説

• 指定した配列変数を消去します。以後同一変数名の配列変数を DIM で宣言することができます。またこのときの配列変数の全要素は 0 またはヌルストリングでクリアされています。

• もし，ERASE を実行せずに同一変数名で宣言すると，“Redimensioned array” エラーが起こります。

• ERASE はその配列変数に割り当てられたメモリ領域を別の目的で使うこともできます。

• ERASE で使用していない配列変数を消去して，空いたメモリ領域を別の目的に使うことができます。このようにして “Out of memory” エラーを防ぐことができます。

サンプルプログラム

```
100 DIM A(100),B(100)
110 ERASE A
120 DIM A(200)
130 DIM B(200) :'ｾﾝｹﾞﾝ ﾃﾞｷﾏｾﾝ !!

run
Redimensioned array in 130
Ok
```

ERL/ERR　N80 N　特殊予約変数

機能　エラーが起こったとき，そのエラーコード及びエラーの発生した行番号を与えます。

書式　**ERL**

ERR

文例　**L=ERL**

E=ERR

解説　• エラーが発生した時点で，ERR はそのエラーコードを，ERL はエラーの発生した行番号を持っています。

• エラーがダイレクトモードの実行によって生じた場合，ERL は行番号として65535を持ちます。

• 通常，ERR 及び ERL は ON ERROR GOTO によって指定したエラー処理ルーチンにおいて，処理の流れを制御するために使われます。

参照　ON ERROR GOTO

サンプルプログラム

```
100 '-- X ノ Y ｼﾞｮｳ ｦ ﾓﾄﾒﾙ｡ --
110 ON ERROR GOTO 200
120 'start
130 INPUT "x,y";X,Y
140 AN=X^Y
150 PRINT X;"ﾉ";Y;"ｼﾞｮｳ ﾊ";ANS;"ﾃﾞｽ｡"
160 GOTO 130
170 PRINT " ??? ﾃｲｷﾞ ｻﾚﾏｾﾝ"
180 END
190 '-- error or routine --
200 'errortop
210 IF ERR=11 AND ERL=140 THEN RESUME 170
220 ON ERROR GOTO 0
run
x,y? 2,3
 2 ﾉ 3 ｼﾞｮｳ ﾊ 8 ﾃﾞｽ｡
x,y? 10,4
 10 ﾉ 4 ｼﾞｮｳ ﾊ 10000 ﾃﾞｽ｡
x,y? -1,3
-1 ﾉ 3 ｼﾞｮｳ ﾊ-1 ﾃﾞｽ｡
x,y? 0,20
 0 ﾉ 20 ｼﾞｮｳ ﾊ 0 ﾃﾞｽ｡
x,y? 0,-1
 ??? ﾃｲｷﾞ ｻﾚﾏｾﾝ
Ok
```

ERROR　N80　N　特殊ステートメント

機能　エラー発生のシミュレート，エラー番号のユーザ定義をします。

書式　**ERROR〈エラー番号〉**

文例　**ERROR 251**

解説

• 〈エラー番号〉の値は1～255までの整数を指定します。

• 指定した値が，すでにBASICでエラーコードとして使われている場合，ERRORが実行されると，〈エラー番号に対応するエラーメッセージを表示し，コマンドレベルに戻ります。また，指定した〈エラー番号〉に対応するエラーメッセージがないときは，“Unprintable Error” と表示されます。

• ON ERROR GOTOがある場合，ERRORが実行されると，エラー処理ルーチンへ分岐します。このとき，ERLにはERRORが実行された行番号，ERRには〈エラー番号〉がそれぞれ代入されます。

• BASICで使用されていない〈エラー番号〉をユーザが定義することによってエラー処理ルーチン内でERRを利用することもできます。

参照　ON ERROR GOTO, ERL/ERR, エラーコード表

サンプルプログラム

```
100 ON ERROR GOTO 180
110 '-- nyuryoku --
120 INPUT "3 ｹﾀ ﾉ ｾｲｽｳ(+) ｦ ﾆｭｳﾘｮｸ ｼﾃｸﾀﾞｻｲ:";N
130 IF FIX(N)<>N THEN ERROR 251
140 IF N<100 OR N>999 THEN ERROR 250
150 PRINT USING "<###>";N
160 END
170 '-- error  message --
180 IF ERR=250 THEN PRINT " 3 ｹﾀ ﾄﾊ 100 ｶﾗ 999 !!!"
190 IF ERR=251 THEN PRINT " ｼﾞｯｽｳ ﾊ ﾆｭｳﾘｮｸ ﾃｾｷﾏｾﾝ "
200 PRINT
210 RESUME 120
run
3 ｹﾀ ﾉ ｾｲｽｳ(+) ｦ ﾆｭｳﾘｮｸ ｼﾃｸﾀﾞｻｲ:? 123456
 3 ｹﾀ ﾄﾊ 100 ｶﾗ 999 !!!

3 ｹﾀ ﾉ ｾｲｽｳ(+) ｦ ﾆｭｳﾘｮｸ ｼﾃｸﾀﾞｻｲ:? 54.3
```

```
 ｼﾞｯｽｳ ﾊ ﾆｭｳﾘｮｸ ﾃﾞｷﾏｾﾝ

3 ｹﾀ ﾉ ｾｲｽｳ(+) ｦ ﾆｭｳﾘｮｸ ｼﾃｸﾀﾞｻｲ:? 999
<999>
Ok
```

EXP [N80] [N] 数値関数

機能 e(2.718281)に対する指数関数の値を与えます。

書式 EXP(〈数式〉)

文例 E=EXP(1)

解説

- 〈数式〉の指数関数の値を与えます。ただし，〈数式〉の値が87.33655より大きい場合には“Over flow”エラーが起こります。
- EXP 関数の演算は単精度で行われます。

サンプルプログラム

```
10 ' EXP
20 '-- Make   SINH(hyperbolic sin)          --
30 '--        COSH(hyperbolic cos) from EXP --
40 INPUT D
50 HS=(EXP(D)-EXP(-D))/2:HC=(EXP(D)+EXP(-D))/2
60 PRINT "SINH";USING "###";D;:PRINT " = ";HS,
70 PRINT "COSH";USING "###";D;:PRINT " = ";HC,
80 END

run
? 2
SINH  2 =  3.62686          COSH  2 =  3.7622
Ok
```

FIELD  入出力ステートメント

機　能　**ランダムファイルのバッファに変数の領域を割り当てます。**

書　式　**FIELD〔#〕<ファイル番号>, <フィールド幅> AS <文字変数>**
〔, <フィールド幅> AS <文字変数>…〕

文　例　**FIELD#1, 128 AS A$, 128 AS B$**

解　説

・ファイルバッファをランダムファイルバッファとして使用するために，ファイルバッファの中に文字変数の領域を割り当てます。

・ランダムファイルの入出力を行う前に必ず，FIELD が実行されていなければなりません。

・<ファイル番号>は OPEN によってファイルをオープンしたときに指定した番号です。また特に<ファイル番号>が0のとき DSKI$, DSKO$ において使用されるランダムバッファを定義します。

・<フィールド幅>は<文字変数>に割り当てる文字数(バイト数)です。バッファの大きさは256バイトなので，各文字変数の<フィールド幅>の合計は256バイト以内でなければなりません。

・1つのバッファに対して，何回かに分けて EIELD を実行することができます。

・FIELD で指定した<文字変数>は必ず LSET/RSET で定義してください。また，この変数を INPUT, LET の左辺で使用しないでください。もし使用すると，その変数は，一般の文字変数領域に登録され，FIELD でのバッファの割り当てが無効となり，正しいファイルの入出力が行われなくなります。

・FIELD で指定した変数は，FIELD を実行した時点で，指定した文字数のヌルキャラクタ(アスキーコードが0のキャラ

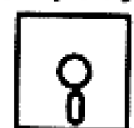
クタ)がセットされます。

参照　OPEN, GET, PUT, LSET/RSET, PC-8001MKIIユーザーズマニュアル

サンプルプログラム

```
100 OPEN "rtest.dat" AS #1
110 FIELD #1,30 AS A$,30 AS B$,30 AS C$
120 '-- DY$ ﾊ A$+B$+C$ ﾉ ﾀﾒﾉ ﾀﾞﾐｰ ﾃﾞｽ｡ --
130 FIELD #1,90 AS DY$,2 AS D$,4 AS E$,8 AS F$
140 LSET A$="NEC-8001MKII
150 LSET B$="Personal"
160 LSET C$="Computer"
170 RSET D$=MKI$(8001)
180 RSET E$=MKS$(1.23456)
190 RSET F$=MKD$(3.14159265358979#)
200 PUT #1,1
210 GET#1,1
220 PRINT A$
230 PRINT B$
240 PRINT C$
250 D%=CVI(D$)
260 E!=CVS(E$)
270 F#=CVD(F$)
280 PRINT D%:PRINT E!:PRINT F#
290 END

run
NEC-8001MKII
Personal
Computer
 8001
 1.23456
 3.14159265358979
Ok
```

FILES/LFILES    一般コマンド

機能 フロッピィディスクに入っているファイルの名前，種類と大きさを表示します。

書式 1) FILE〔<ドライブ番号>〕

2) LFILES〔<ドライブ番号>〕

文例 FILES 2

解説 ・<ドライブ番号>で指定された，フロッピィディスク上に登録されているファイルの名前，種類とその大きさを表示します。

・書式1)の場合ディスプレイ画面に，2)の場合プリンタに出力します。

・<ドライブ番号>が省略されたときは，ドライブ1が選択されます。

・ファイルの種類は，出力される<ファイル名>と<拡張子>との区切り文字によって表わされます。区切り記号が空白の場合には，そのファイルはアスキーセーブされたプログラムファイル，または，OPENによって作られたデータファイルであることを示します。区切り文字が，ピリオドの場合には，バイナリセーブによって作られたプログラムファイル，アスタリスクの場合にはCMD BSAVEによって作られた機械語プログラムファイルであることを表します。

・ファイルの大きさは，クラスタを単位として表示されます。ミニフロッピィディスクの1クラスタは8セクタ，8インチフロッピィディスクの1クラスタは26セクタです。

参照 SAVE, CMD BSAVE

サンプル
プログラム

```
files
format.       1     backup.       1
gpib  .       1     table .       7
starfi.re     7     timebo.mb     3
3DM   .       4     ﾀｯﾄ ｹﾞ.ｰﾑ     4
datage.ne     1     bascom.       1
reviva.l      1     file  .       5
INVADE.R      10    galawa.r      2
alien .fal    6     othell.o      5

Ok
```

FIX N80 N 数値関数

機能　整数部を与えます。

書式　FIX(<数式>)

文例　A=FIX(B/5)

解説
• <数式>の値の小数点以下を取り去った値を与えます。

• FIX（<数式>）とINT（<数式>）の値は<数式>の値が正のとき等しくなりますが，負となった場合FIX関数は小数部分を取り去ってしまいます。したがってこの場合はFIX関数で得られる値がINT関数で得られる値よりも大きくなります。

参照　INT

サンプルプログラム

```
100 '-- fix & int --
110 A=-2
120 FOR I=0 TO 9
130 A=A+I*.3
140 PRINT "A =";A,"FIX(A) =";FIX(A),"INT(A) =";INT(A)
150 NEXT I
run
A =-2          FIX(A) =-2    INT(A) =-2
A =-1.7        FIX(A) =-1    INT(A) =-2
A =-1.1        FIX(A) =-1    INT(A) =-2
A =-.2         FIX(A) = 0    INT(A) =-1
A = 1          FIX(A) = 1    INT(A) = 1
A = 2.5        FIX(A) = 2    INT(A) = 2
A = 4.3        FIX(A) = 4    INT(A) = 4
A = 6.4        FIX(A) = 6    INT(A) = 6
A = 8.8        FIX(A) = 8    INT(A) = 8
A = 11.5       FIX(A) = 11   INT(A) = 11
Ok
```

FOR…TO…STEP～NEXT　N80 N

一般ステートメント

機　能　FORとNEXTにはさまれた一連の命令を指定回数だけ繰り返し実行します。

書　式

```
FOR〈変数〉=〈初期値〉 TO 〈終値〉〔STEP〈増分〉〕
 ～
NEXT〔〈変数〉〔,〈変数〉…〕〕
```

文　例

```
FOR I=0 TO 50 STEP 2
 ～
NEXT I
```

解　説

- 〈変数〉は制御変数と呼ばれ，ループのカウンタとして使われます。
- 〈変数〉は整数変数，単精度変数のみ使用できます。文字変数，倍精度変数は使用できません。
- 〈初期値〉，〈終値〉，〈増分〉は数式です。
- FOR～NEXT の動作は次のようになります。

まず〈変数〉に〈初期値〉が設定され，FOR 以後からそれに対応する NEXT まで実行されます。そして STEP によって指定された〈増分〉を〈変数〉に加え,それを新しい〈変数〉とします。次に〈変数〉の値が〈終値〉と比べられ，〈終値〉に達していなければ，FOR の次の文へ戻り，同じ処理が操り返されます。これを FOR～NEXT ループと呼びます。STEP が省略された場合〈増分〉は＋１とみなされます。また〈増分〉は負の値をとることもできます。このとき〈初期値〉は，〈終値〉より大きくしなければなりません。この場合〈変数〉が〈終値〉よりも小さくなるまで FOR～NEXT ループが操り返されます。

- 次の場合FOR～NEXTループは１回だけ実行されます。

　1）〈増分〉が正の値で,〈初期値〉が〈終値〉より大きい場合。

2） 〈増分〉が負の値で，〈初期値〉が〈終値〉より小さい場合。

3） 〈初期値〉と〈終値〉が同じ値で〈増分〉が 0 の場合。

但し，〈変数〉には〈初期値〉が代入されています。

• FOR～NEXT ループは入れ子構造にすることができます。これは 1 つの FOR～NEXT ループの中にもう 1 つの FOR～NEXT ループを置くことができるということです。このとき，それぞれの〈変数〉は別のものを使わなければなりません。

• 1 つの FOR に対して，入れ子構造さえ正しければ，NEXT はいくつあってもかまいません。

サンプルプログラム

```
100 CMD SCREEN 3,,6:CMD CLS 3
110 FOR R=10 TO 200 STEP 20
120  FOR I=0 TO 3.14 STEP .05
130     Y=SIN(I)*COS(I)
140     X=SIN(I)*SIN(I)
150     Y=Y*R+100
160     X=(X*R*2.5+100-C)/2
170     CMD PSET(X,Y)
180   NEXT I
190   C=C+10
200 NEXT R
210 END
```

FORMAT ディスク N 特殊コマンド

(機　能)　ミニフロッピィディスクを初期化(フォーマット)します。

(書　式)　**FORMAT〔<ドライブ番号>〕**

(文　例)　**FORMAT**

(解　説)

• FORMAT は, DISK-BASICを使用する場合には通常使う必要はありません。逆に FORMAT によって重要なデータを消してしまう危険性もありますから使わない方がよいでしょう。フロッピィディスクを初期化する場合は, システムディスクに入っている "format" というユーティリティプログラムを使用してください。

• FORMAT は, ミニフロッピィディスクの初期化を行います。しかし, FORMAT を実行したフロッピィディスクは, FAT , ディレクトリ, ID の初期化は行われていませんから DISK-BACIC 用に使うことはできません。DISK-BACIC から実行できるのは DSKI$, DSKO$ だけです。

(注　意)

1. FORMAT を実行するとそのフロッピィディスクに書き込まれていたファイルはすべて消去されます。したがって, 重要なファイルが書き込まれているフロッピィディスクに対して FORMAT をしないように注意してください。
2. FORMAT が使用できるのはインテリジェントタイプのミニフロッピィディスクユニットだけです。
3. N80 DISK-BASIC には FORMAT に相当する命令は用意されていません。フロッピィディスクを初期化するには, システムディスクに入っている "formatN80" を使用してください。

(サンプルプログラム)

```
format 2
Ok
```

FPOS　ディスク ［ディスク］［N80］［N］　入出力関数

機能　ファイル中での物理的な現在位置を与えます。

書式　FPOS(<ファイル番号>)

文例　HD=FPOS(2)

解説　•<ファイル番号>によって指定されたファイルが最後に読み書きしたフロッピィディスク上の物理的な位置を返します。この番号は，トラック0，ヘッド0，セクタ1を0番としたときの通し番号で与えられます。

参照　LOC

サンプルプログラム

```
100 '-- fpos value print out --
110 OPEN "2:data6" FOR OUTPUT AS#1
120 PRINT FPOS(1):PRINT
130 FOR I=1 TO 5
140   PRINT #1,STRING$(128,"1")
150   PRINT #1,STRING$(128,"1")
160   PRINT FPOS(1)
170 NEXT I
180 CLOSE #1
run
 120

 122
 123
 124
 125
 126
Ok
```

FRE　N80　N　特殊関数

機能　**メモリの未使用領域の大きさを与えます。**

書式　**FRE**(〈**機能**〉)

文例　**PRINT FRE(0)**

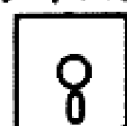
解説

• 〈機能〉が数値型であればメモリの未使用領域の大きさを，文字型であれば未使用の文字領域の大きさをバイト数で返します。

サンプルプログラム

```
100 CLEAR 300,&HD000
110 PRINT "ﾌﾟﾛｸﾞﾗﾑ ﾘｮｳｲｷ ﾊ";FRE(0) "byte free ﾃﾞｽ"
120 PRINT "ﾓｼﾞﾚﾂ    ﾘｮｳｲｷ ﾊ";FRE("a")"byte free ﾃﾞｽ"

run
ﾌﾟﾛｸﾞﾗﾑ ﾘｮｳｲｷ ﾊ 20024 byte free ﾃﾞｽ
ﾓｼﾞﾚﾂ    ﾘｮｳｲｷ ﾊ 300 byte free ﾃﾞｽ
Ok
```

GET    入出力ステートメント

(機能) **ランダムアクセスファイルのデータをバッファに読み込みます。**

(書式) **GET〔#〕〈ファイル番号〉〔,〈レコード番号〉〕**

(文例) **GET #3, 5**

(解説) • 〈ファイル番号〉で指定されたファイル中のデータを，対応するバッファの中に読み込みます。指定されたファイルはランダムアクセスのモードでオープンされていなければなりません。

• 〈レコード番号〉はファイルのレコード番号で，指定されたレコード(256バイト)がバッファに読み込まれます。レコード番号が省略された場合には，直前に行われた，GET, PUTで指定されたレコードの次のレコードが読み込まれます。

• GET を実行した後 INPUT #, LINE INPUT # 等で同一の〈ファイル番号〉を指定すると，そのバッファに入っているデータから入力が行われますので注意してください。

• GET はバッファの読み込みを行なうものです。変数にGET したデータを割り付けるには，FIELD を用います。

(参照) OPEN, FIELD, PUT, PC-8001MKII ユーザーズマニュアル第8章

(サンプルプログラム)

```
100 '-- random data read --
110 OPEN "1:data1" AS #1
120 FIELD #1,30 AS A$,20 AS B$,50 AS C$
130 GET #1,1
140 PRINT "NAME",A$
150 PRINT "TEL",B$
160 PRINT "ADDRESS",C$
170 CLOSE
run
NAME          ｱﾗｲ ﾏｻﾖ
TEL           03-876-5432
ADDRESS       ﾄｳｷｮｳ ﾄ ｼﾌﾞﾔ ｸ ｱｵﾔﾏ 3 ﾁｮｳﾒ
Ok
```

GET@　N80 N　グラフィックステートメント

機能　**テキスト画面上の任意の範囲のキャラクタまたは，低解像度グラフィックパターンを配列へセーブします。**

書式
1）GET〔@〕(X_1, Y_1)－(X_2, Y_2)，〈配列名〉
2）GET〔@〕(Lx_1, Ly_1)－(Lx_2, Ly_2)，〈配列名〉，G
3）GET〔@〕A (X_1, Y_1)－(X_2, Y_2)，〈配列名〉

文例
1）GET@(10, 10)－(35, 19)，A%
2）GET@(5, 5)－(50, 60)，B%, G
3）GET@(5, 5)－(14, 14),C%

解説
1） キャラクタ座標において(X_1, Y_1)と(X_2, Y_2)を対角線とする四角形の内側に表示されているキャラクタを配列へセーブします。この場合の水平位置X_1, X_2は０～(１行の桁数)－１までの値，垂直位置Y_1, Y_2は０～(１画面の行数)－１までの値で指定し，配列の添字の値は次のようになります。

〈添字の値〉＝(キャラクタ単位で計った面積/N)－1

Nは整数型配列で２，実数型配列で４，倍精度型配列で８となります。

2） 低解像度グラフィック座標において(Lx_1, Ly_1)と(Lx_2, Ly_2)を対角線とする四角形の内側に表示されている低解像度グラフィックパターンを配列にセーブします。この場合の水平位置Lx_1, Lx_2は０～(１行の桁数)×２－１までの値，垂直位置Ly_1, Ly_2は０～(２画面の行数)×４－１までの値をとります。配列の添字の値は次のようになります。

〈添字の値〉＝(ピクセル単位で計った面積／８＋２)/N

Nは先程のキャラクタの場合と同じです。

3) キャラクタと低解像度グラフィックパターン，あるいは色や機能の異なる画面上のデータを配列にセーブします。座標の指定は書式1)の場合と同じで，配列の添字の値は次のようになります。

〈添字の値〉＝（キャラクタ単位で計った面積/N）＊2−1

このときのNも先程と同じです。

参照　PUT @

サンプルプログラム

```
100 WIDTH 80,25
110 CONSOLE ,,,1
120 COLOR 4,0,1
130 DIM G%(100)
140 LINE(0,0)-(10,10),PSET,BF
150 GET@(0,0)-(10,10),G%,G
160 PUT@(100,50)-(110,60),G%,PSET
170 END
```

GOSUB～RETURN　N80 N　一般ステートメント

機能　**サブルーチンのコール，及びサブルーチンからのリターンを行います。**

書式　**GOSUB<行番号>**

文例　**GOSUB 1000**

解説　• <行番号>から始まるサブルーチンプログラムを GOSUB によってコールし，サブルーチンプログラム内の RETURN によって，GOSUB の次の文へリターンします。

• サブルーチンプログラムとは，独立したプログラムで，RETURN で終っているものをいいます。これらは GOSUB によって，いつでも何度でもコールすることができます。

• 1つのサブルーチンの中から他のサブルーチンをコールすることもでき，これをサブルーチンの多重化(ネスティング)と呼びます。この多重化はメモリのスタック領域の容量が許す限り行うことができます。もし，スタック領域が足りなくなった場合には "Out of memory" エラーが起こります。

• サブルーチンは必ず GOSUB によってコールされなければなりません。もし，単独で実行した場合，RETURN に出会うと，"RETURN without GOSUB" エラーが起ります。

• 1つのサブルーチン内に複数の RETURN があってもかまいませんが，正しく GOSUB と対応していなければなりません。

注意　"GO" と "SUB" の間にスペースを書いた場合，BASIC はそれを GOSUB とは解釈しません。

サンプルプログラム

```
100 '-- ｻﾝｶｸｹｲ ﾉ ﾒﾝｾｷ ｦ ﾓﾄﾒﾙ --
110 '-- start --
120 INPUT "ﾃｲﾍﾝ :";BA
130 INPUT "ﾀｶｻ  :";H
140 GOSUB 190
150 PRINT "ﾒﾝｾｷ ﾊ";AR
160 PRINT
170 GOTO 120
180 '-- menseki --
190 AR=BA*H/2
200 RETURN

run
ﾃｲﾍﾝ :? 78
ﾀｶｻ  :? 8
ﾒﾝｾｷ ﾊ 312

ﾃｲﾍﾝ :? 100
ﾀｶｻ  :? 30
ﾒﾝｾｷ ﾊ 1500

ﾃｲﾍﾝ :? 10000
ﾀｶｻ  :? 7890
ﾒﾝｾｷ ﾊ 3.945E+07

ﾃｲﾍﾝ :?
Break in 120
Ok
```

GOTO/GO TO　N80 N　一般ステートメント

機能　指定した行へ無条件にジャンプします。

書式　1)　GOTO<行番号>

2)　GO TO<行番号>

文例　GOTO 500

解説　• <行番号>で指定した行へ無条件にジャンプします。書式1)，2)は全く同じ機能です。

注意　"GO" と "TO" の間に2個以上のスペースを書いた場合 BASIC はそれを GOTO とは解釈しません。

サンプルプログラム

```
100 '-- start --
110 PRINT "How do you do?"
120 GOTO 110

run
How do you do?
How do you do?
How do you do?
How do you do?
How do you do?
How do you do?
How do you do?
How do you do?
^C
Break in 110
Ok
```

HEX$ N80 N 文字関数

機能 **10進数を16進数の文字列に変換します。**

書式 **HEX$**(〈数式〉)

文例 **PRINT HEX$**(**H**)

解説 • 〈数式〉の値を16進数に変換して，その文字列を与える関数です。

• 〈数式〉の値は，－32768～32767(または０～65535)までです。また，〈数式〉に小数点が含まれる場合，小数点以下を切捨てます。

サンプルプログラム

```
100 '-- 10 ｼﾝ <--> 16 ｼﾝ ﾍﾝｶﾝ ﾋｮｳ --
110 PRINT "     -0 -1 -2 -3 -4 -5 -6 -7 -8 -9
120 FOR I=0 TO 9
130   PRINT STR$(I);"- ";
140   FOR J=0 TO 9
150     PRINT RIGHT$("0"+HEX$(I*10+J),2);" ";
160   NEXT J
170   PRINT
180 NEXT I
190 END
run
     -0 -1 -2 -3 -4 -5 -6 -7 -8 -9
 0- 00 01 02 03 04 05 06 07 08 09
 1- 0A 0B 0C 0D 0E 0F 10 11 12 13
 2- 14 15 16 17 18 19 1A 1B 1C 1D
 3- 1E 1F 20 21 22 23 24 25 26 27
 4- 28 29 2A 2B 2C 2D 2E 2F 30 31
 5- 32 33 34 35 36 37 38 39 3A 3B
 6- 3C 3D 3E 3F 40 41 42 43 44 45
 7- 46 47 48 49 4A 4B 4C 4D 4E 4F
 8- 50 51 52 53 54 55 56 57 58 59
 9- 5A 5B 5C 5D 5E 5F 60 61 62 63
Ok
```

IF…THEN…ELSE/IF…GOTO…ELSE　N80　N

一般ステートメント

機能　**論理式の条件判断を行い次に実行する文を選択します。**

書式

IF〈論理式〉 | THEN |〈文〉|〈行番号〉| | GOTO〈行番号〉 | 〔ELSE |〈文〉|〈行番号〉| 〕

文例　**IF A$="y" THEN 200 ELSE END**

解説

• 論理式の条件によってプログラムの実行を制御します。

• 〈論理式〉が真(0以外)ならばTHENまたはGOTO以降を実行し，偽(0)ならばELSE以降が実行されます。もしELSE以降が省略されている場合，次の行から実行されます。

• IF…THEN…ELSEにおいて，THENやELSEに続けて別のIFを置いて多重にすることができます。もしIFの中のTHENとELSEの個数が異なるときELSEは，最も近くにあり他のELSEと対応していないTHENに対応します。但し，多重できるのは1行(255文字に書ける範囲)に限られます。

サンプルプログラム

```
100 '-- real or imaginary --
110 INPUT A
120 FL=0
130 IF A<0 THEN A=ABS(A):FL=1
140 PRINT SQR(A);
150 IF FL THEN PRINT "i" ELSE PRINT
160 END
```


```
100 '-- real or imaginary --
110 INPUT A
120 FL=0
130 IF A<0 THEN A=ABS(A):FL=1
140 PRINT SQR(A);
150 IF FL THEN PRINT "i" ELSE PRINT
160 END
```


```
run
? 5
 2.23607
Ok
run
? -98
 9.8995 i
Ok
run
? -25
 5 i
Ok
```

INIT%　N80 N　入出力ステートメント

機能　RS-232C 通信ポートを初期設定(イニシャライズ)します。

書式　INIT%<ポート番号>, <モード>, <コマンド>

文例　INIT% 1, &HFB, &H37

解説

・<ポート番号>で指定された RS-232C 通信ポートの入出力バッファのイニシャライズを行います。MKⅡ本体に内蔵されている RS-232C インタフェースを使用する場合，<ポート番号>は 1 を指定します。

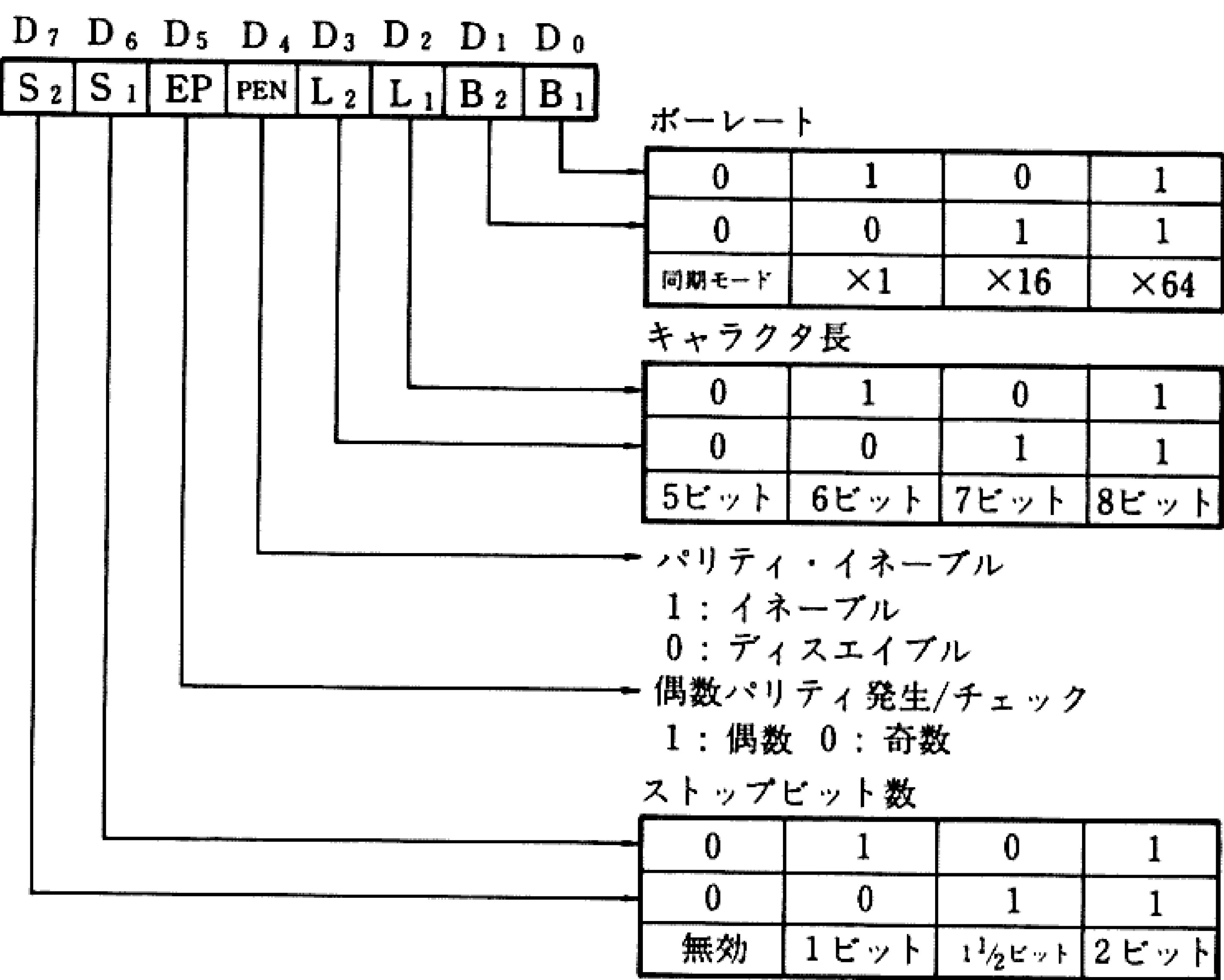
・<モード>と<コマンド>はプログラマブル・コミュニケーションインタフェース μPD8251C の動作を指定します。

・イニシャライズを行っていない状態で RS-232C インタフェースを使用すると，"Port not initialized" エラーが起こります。

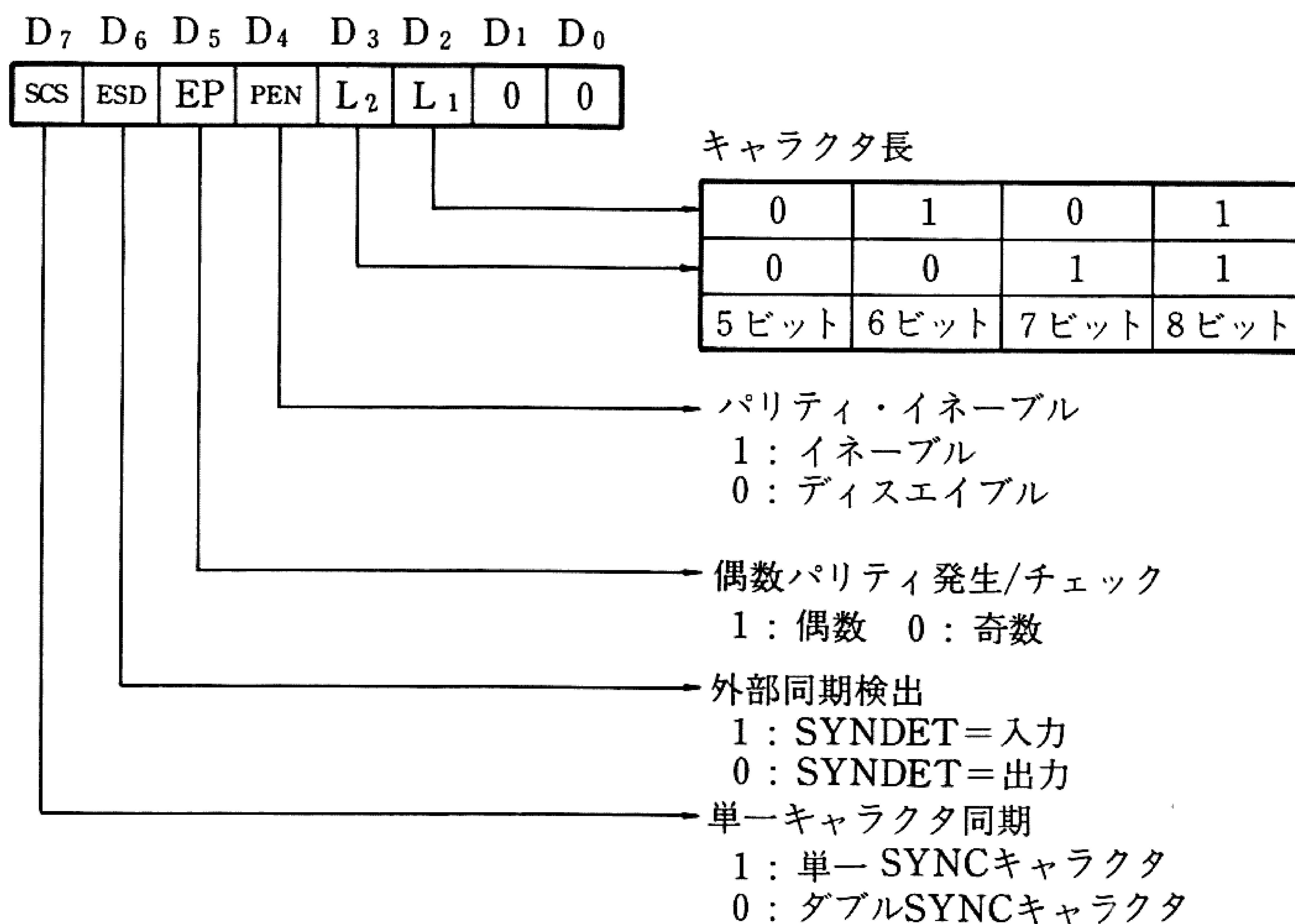
・μPD8251C の<モード>と<コマンド>は次のようにそれぞれ 8 ビットのデータを設定します。

<モード>

非同期モード

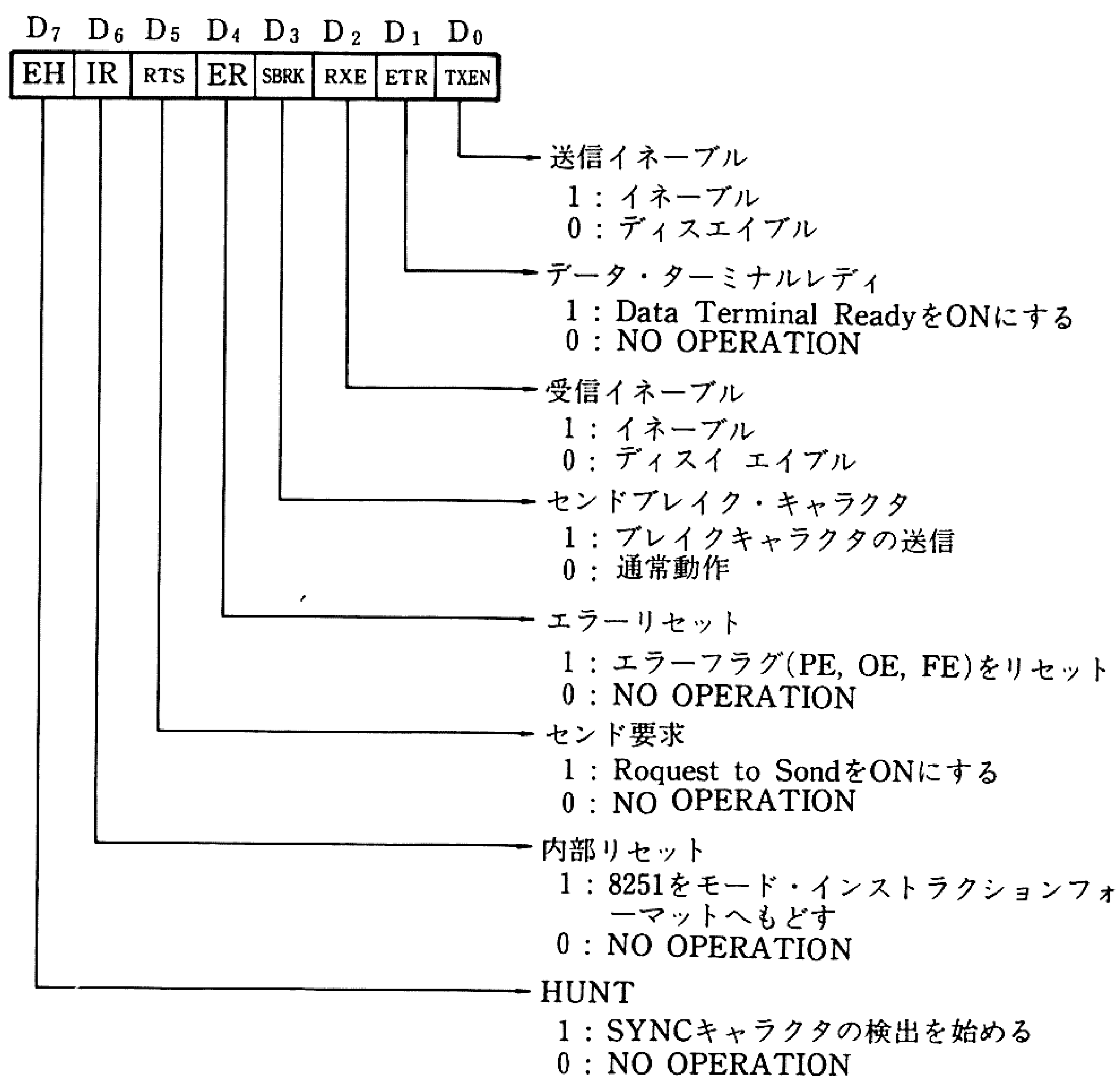


同期モード



D7 D6 D5 D4 D3 D2 D1 D0
SCS ESD EP PEN L2 L1 0 0
キャラクタ長
0 1 0 1
0 0 1 1
5ビット 6ビット 7ビット 8ビット
パリティ・イネーブル
1：イネーブル
0：ディスエイブル
偶数パリティ発生/チェック
1：偶数　0：奇数
外部同期検出
1：SYNDET＝入力
0：SYNDET＝出力
単一キャラクタ同期
1：単一SYNCキャラクタ
0：ダブルSYNCキャラクタ

〈コマンド〉



D7 D6 D5 D4 D3 D2 D1 D0
EH IR RTS ER SBRK RXE ETR TXEN
送信イネーブル
1：イネーブル
0：ディスエイブル
データ・ターミナルレディ
1：Data Terminal ReadyをONにする
0：NO OPERATION
受信イネーブル
1：イネーブル
0：ディスイ エイブル
センドブレイク・キャラクタ
1：ブレイクキャラクタの送信
0：通常動作
エラーリセット
1：エラーフラグ(PE, OE, FE)をリセット
0：NO OPERATION
センド要求
1：Roquest to SondをONにする
0：NO OPERATION
内部リセット
1：8251をモード・インストラクションフォーマットへもどす
0：NO OPERATION
HUNT
1：SYNCキャラクタの検出を始める
0：NO OPERATION

サンプルプログラム

```
100 ´jump table set
110 POKE &H8001,&H23:POKE &H8000,&H7F
120 ´interrupt enable
130 OUT &HE6,4
140 ´current status register set
150 OUT &HE4,&HFF
160 ´ﾓｰﾄﾞﾊﾞｲﾄ :ｽﾄｯﾌﾟ ﾋﾞｯﾄ 2
170 ´          :ｸﾞｳｽｳ ﾊﾟﾘﾃｨ
180 ´          :ﾊﾟﾘﾃｨ ｲﾈｰﾌﾞﾙ
190 ´          :ｷｬﾗｸﾀﾁｮｳ  7
200 ´          :ﾎﾞｰ ﾚｲﾄ ﾌｧｸﾀ *16
210 ´ｺﾏﾝﾄﾞﾊﾞｲﾄ:ｾﾝﾄﾞ ﾖｳｷｭｳ ON
220 ´          :ｴﾗｰﾘｾｯﾄ
230 ´          :ｼﾞｭｼﾝ ｲﾈｰﾌﾞﾙ
240 ´          :ﾃﾞｰﾀ ﾀｰﾐﾅﾙ ﾚﾃﾞｨ
250 ´          :ｿｳｼﾝ ｲﾈｰﾌﾞﾙ
260 INIT %1,&HFA,&H37
270 IF PORT(1) THEN PRINT INPUT$(PORT(1),%1);
280 C$=INKEY$:IF C$<>"" THEN PRINT %1,C$;
290 GOTO 270
300 ´********************************
310 ´ ﾌﾟﾛｸﾞﾗﾑ ｼｭｳﾘｮｳ ｺﾞ out &hE6,&h0
320 ´ ｦ ｼﾞｯｺｳ ｼﾃ ｸﾀﾞｻｲ｡
330 ´********************************
```

INKEY$ N80 N 文字関数

機能　キーが押されていれば，その文字を，押されていなければヌルストリングを与えます。

書式　**INKEY$**

文例　**A$=INKEY$**

解説　• キー入力があれば，その文字を返します。ただし，キー入力された文字は画面に表示されません。

• CTRL + C と STOP キーは INKEY$ 関数によって拾うことはできません。

サンプルプログラム

```
100 PRINT "Enter any key to end .."
110 'loop
120 A$=INKEY$
130   IF A$="" THEN 120
140   PRINT A$
150   IF A$="e" THEN 160 ELSE 120
155 'exit
160 END

run
Enter any key to end ..
A
b
c
D
E
e
Ok
```

INP　N80 N　特殊関数

機能　入力ポートから値を得ます。

書式　INP(<I/Oアドレス>)

文例　A=INP(0)

解説　• <I/Oアドレス>で指定された入力ポートから8ビットのデータを読み込み，それを関数の値とします。

• <I/Oアドレス>として指定できる値の範囲は0～255までです。

参照　OUT, PC8001MKⅡユーザーズマニュアル付録9 I/Oマップ

サンプルプログラム

```
100 PRINT "Type 'e' key to end .."
110 'loop
120 IF INP(2)=223 THEN END
130 DM$=INPUT$(1):PRINT DM$
140 GOTO 120
run
Type 'e' key to end ..
a
b
c
d
e
Ok
```

INPUT　N80 N　入出力ステートメント

機能　**キーボードより入力したデータを指定した変数に代入します。**

書式　**INPUT〔<"プロンプト文"> ; 〕<変数名>〔, <変数名>…〕**

文例　**INPUT "DATA" ; A$**

解説

• INPUT 文が実行されると，画面に疑問符(？)と1桁のスペースを表示し，キーボードからの入力待ちの状態になります。このとき<"プロンプト文">がある場合,<"プロンプト文">に続いて疑問符を表示します。

• データは RETURN キーを押すことによって入力され，変数に代入されます。<変数名>をコンマで区切って複数個指定した場合には，入力するデータもコンマで区切って<変数名>の数だけ入力するか, 1つずつ RETURN キーを押して入力します。この場合， 2回目からは"？？"が表示され入力待ちとなります。

• <変数名>と入力されたデータとの型と数が一致しなかったときには，"Redo from start" と表示し，再び入力待ちになります。また，データが<変数>の個数より多い場合には "Extra ignored" と表示し，次の文から実行が続けられます。

• RETURN キーだけを押した場合，変数は現在の値が採用されます。したがって INPUT を実行する直前と変わりません。

• 文字変数に文字列を代入する場合，コンマや文字列の前後の意味のある空白を入力したいときは，ダブルクォーテーションマーク(")で文字列を囲む必要があります。この場合は文字列として引用符を入力することはできません。

サンプルプログラム

```
100 INPUT "ﾅﾏｴ ";NA$
110 INPUT "ｾｲﾍﾞﾂ ";SE$
120 INPUT "ﾈﾝﾚｲ ";OD
130 PRINT "ﾅﾏｴ    :";NA$
140 PRINT "ｾｲﾍﾞﾂ :";SE$
150 PRINT "ﾈﾝﾚｲ  :";OD;"ｻｲ"
```

```
run
ﾅﾏｴ ? ﾜﾀﾅﾍﾞｸﾝ
ｾｲﾍﾞﾂ ? ｵﾄｺ
ﾈﾝﾚｲ ? ﾊﾀﾁ
?Redo from start
ﾈﾝﾚｲ ? 20
ﾅﾏｴ    :ﾜﾀﾅﾍﾞｸﾝ
ｾｲﾍﾞﾂ :ｵﾄｺ
ﾈﾝﾚｲ  : 20 ｻｲ
Ok
```

INPUT＃ N80 N 入出力ステートメント

機　能　**シーケンシャルファイルからデータを読み込みます。**

書　式　**INPUT＃〈ファイル番号〉，〈変数〉〔,〈変数〉…〕**
〈ファイル番号〉が1〜15の場合は ディスク

文　例　**INPUT＃1, A, B**

解　説　• INPUT がキーボードからデータを読み込んだのに対し，INPUT ＃は，フロッピィディスクとカセットテープに書き込まれたシーケンシャルファイルからデータを読み込みます。

• フロッピィディスクの場合，〈ファイル番号〉は，対象となるシーケンシャルファイルを，OPEN によってオープンしたときに指定した〈ファイル番号〉と一致しなければなりません。このとき〈ファイル番号〉は，1〜15の値をとります。

• カセットテープでは，ファイルをオープンすることは必要ありません。この場合〈ファイル番号〉は－1に固定されています。

• INPUT ＃で読み込むデータは，PRINT ＃，PRINT ＃ USING で書き込んだものです。INPUT ＃の〈変数〉の並びは，書き込み時のデータの形と一致していなければなりません。

参　照　OPEN, PRIN ＃, PRINT ＃ USING, 付録A, PC-8001MKII ユーザーズマニュアル

サンプルプログラム

```
100 '-- sequential data read --
110 OPEN "stest.dat" FOR INPUT AS#1
120 'input data
130 IF EOF(1) THEN END
140 INPUT #1,A$
150 PRINT A$
160 GOTO 130
170 END

run
CAT
DOG
HORSE
SHEEP
MOUSE
Ok
```

INPUT$ [N80] [N] 入出力関数

(機能) キーボード，ファイル，または，バッファより指定された長さの文字を与えます。

(書式) INPUT$(〈文字数〉〔, |〔#〕〈ファイル番号| 〕)
　　　　　　　　　　　　　|%〈ポート番号〉|

ファイル番号が1～15の場合は ディスク

(文例) CM$=INPUT$(6)

(解説)

- 〈文字数〉は0～255の範囲でなければなりません。
- 〈ファイル番号〉が省略された場合には，キーボードから指定された〈文字数〉だけ読み込みます。このとき入力された文字は画面に表示されません。
- 〈ファイル番号〉が指定されると，ファイルから〈文字数〉分の文字列を読み出します。また，〈ポート番号〉を指定するとRS-232Cのバッファより〈文字数〉分の文字列を読み出します。
- INPUT$は指定された〈文字数〉の文字が入力されるのを待ち続けますが，既にバッファに入力済みのデータがある場合には，バッファの先頭から文字を拾ってきます。またINPUT$は，STOP キー，CTRL ＋ C を除くすべての文字をそのまま読み出しますので，INPUTやLINE INPUTでは入力することのできない改行文字等も入力することができます。

(参照) INPUT, LINE INPUT

(サンプルプログラム)

```
100 '-- get key --
110 FOR I=1 TO 5
120   A$=INPUT$(I)
130   PRINT A$
140 NEXT I
150 END
```

```
run
a
ab
abc
abcd
abcde
Ok
```

INPUT% Ⓝ80 Ⓝ 入出力ステートメント

機能 RS-232C通信ポートの入出力バッファからデータを入力します。

書式 **INPUT%〈ポート番号〉，〈変数名〉〔,〈変数名〉…〕**

文例 **INPUT%1, A$**

解説

• 〈ポート番号〉で指定されたRS-232C通信ポートの入出力バッファよりデータを入力し，その値を〈変数名〉で指定される変数に代入します。このときMKⅡ内蔵のRS-232Cインタフェースを使用する場合，〈ポート番号〉は1を指定します。

• 入力されるデータはCR, LFで区切られます。バッファの先頭からこの区切りをさがし，区切りが見つかったところで，それらを変数に代入します。もし，バッファ内にCR, LFがなければRS-232Cチャネルを通してこの区切りが入力されるまで待ちます。

参照 PRINT %, INIT %, PORT

サンプルプログラム

```
100 'input % test
110 POKE &H8001,&H23:POKE &H8000,&H7F
120 OUT &HE6,4
130 OUT &HE4,&HFF
140 INIT %1,&HFA,&H37
150 INPUT %1,A$
160 IF A$="end" THEN OUT &HE6,0:END
170 PRINT A$
180 GOTO 150
```

INSTR　N80 N　文字関数

機能　**文字列の中から任意の文字列を捜し，その文字の位置を与えます。**

書式　**INSTR(〔〈数式〉,〕〈文字列 1〉, 〈文字列 2〉)**

文例　**A=INSTR(A$, "cmd")**

解説

- 〈文字列 1〉の中から〈文字列 2〉を探し，見つかればその位置を与えます。このとき〈数式〉は捜し始める位置を指定し，省略した場合は，〈文字列 1〉の先頭から探し始めます。
- 〈文字列 2〉が見つからなかったとき，〈文字列 1〉がヌルストリングであったとき，〈数式〉が〈文字列 1〉の長さより大きい場合 0 を値として返します。
- 〈文字列 2〉にヌルストリングを指定すると，〈数式〉と同じ値を返します。

サンプルプログラム

```
100 '-- alphabet search --
110 A$="abcdefghijklmnopqrstuvwxyz"
120 AL$=INPUT$(1)
130 AL=ASC(AL$)
140   IF AL>64 AND AL<91 THEN AL=AL+32
150   A=INSTR(A$,CHR$(AL))
160   IF A=0 THEN PRINT AL$;" ﾊ ｱﾙﾌｧﾍﾞｯﾄ ﾃﾞﾊ ｱﾘﾏｾﾝ":END
170   PRINT USING "! ﾊ ## ﾊﾞﾝﾒ ﾉ ｱﾙﾌｧﾍﾞｯﾄﾃﾞｽ";AL$,A
180 END

run
a ﾊ  1 ﾊﾞﾝﾒ ﾉ ｱﾙﾌｧﾍﾞｯﾄﾃﾞｽ
Ok
run
z ﾊ 26 ﾊﾞﾝﾒ ﾉ ｱﾙﾌｧﾍﾞｯﾄﾃﾞｽ
Ok
run
9 ﾊ ｱﾙﾌｧﾍﾞｯﾄ ﾃﾞﾊ ｱﾘﾏｾﾝ
Ok
```

INT　N80 N　数値関数

機能　**小数点以下を切り捨てた整数値を与えます。**

書式　**INT(<数式>)**

文例　**A=INT(1.23)**

解説

- <数式>の値を超えない最大の整数を与えます。
- FIX との違いに注意してください。

参照　FIX, CINT

サンプルプログラム

```
100 '-- ｾｲｽｳ ﾄ ｼﾞｯｽｳ ﾉ ｸﾍﾞﾂ --
110 DIM A$(1)
120 A$(0)="ｼﾞｯｽｳ ﾃﾞｽ"
130 A$(1)="ｾｲｽｳ ﾃﾞｽ"
140 FL=0
150 INPUT A
160 IF A=INT(A) THEN FL=1
170 PRINT A;" ﾊ ";A$(FL)
180 END

run
? 123
 123  ﾊ ｾｲｽｳ ﾃﾞｽ
Ok
run
? 123.456
 123.456  ﾊ ｼﾞｯｽｳ ﾃﾞｽ
Ok
```

KEY [N80] [N] 一般ステートメント

機能 キーボード上にあるプログラマブル・ファンクションキーの内容を定義します。

書式 KEY<キー番号>, <文字列>

文例 KEY 1, "CMD COPY"+CHR$(13)

解説

• ファンクションキーは5つ [f･1] ～ [f･5] あり，シフトキーを押したときのファンクションキーを加えると合計10の<文字列>を記憶させておくことができます。

• 各ファンクションキーは最大15文字までの文字列およびコントロール文字を定義することができます。

• キーボードから直接入力できない文字は，CHR$関数でつないで使います。

• <キー番号>は1～10までで指定してください。

参照 KEYLIST, CHR$

サンプルプログラム

```
100 KEYLIST
110   FOR I=1 TO 10
120    KEYI,STRING$(15,CHR$(64+I))
130   NEXT I
140 KEYLIST

run

cmd                time$
auto               key
go to              print
list               list.
run                cont

AAAAAAAAAAAAAAA FFFFFFFFFFFFFFF
BBBBBBBBBBBBBBB GGGGGGGGGGGGGGG
CCCCCCCCCCCCCCC HHHHHHHHHHHHHHH
DDDDDDDDDDDDDDD IIIIIIIIIIIIIII
EEEEEEEEEEEEEEE JJJJJJJJJJJJJJJ
Ok
```

KEY LIST　N80 N　一般コマンド

機　能　ファンクションキーの内容を画面にリストアウトします。

書　式　**KEY LIST**

文　例　**KEY LIST**

解　説　• ファンクションキーの内容の一部は画面の最下行に表示させておくことができますが，KEY LIST は，そのすべての内容を画面に表示させることができます。

参　照　KEY

サンプルプログラム

```
keylist

cmd               time$
auto              key
go to             print
list              list.
run               cont
Ok
run
```

KILL　ディスク N80 N　一般コマンド

機能　フロッピィディスク上のファイルを削除します。

書式　KILL<ファイル名>

文例　KILL "2 : test"

解説

• <ファイル名>で指定したファイルをフロッピィディスクから削除します。

• 削除するファイルはクローズされていなければなりません。もし，オープン状態のファイルを削除しようとすると "File already open" エラーが起ります。また，書き込み禁止属性の付けられたファイルを KILL で削除しようとすると "File write protected" エラーが起ります。この場合 SET で書き込み禁止属性を解除してから KILL を実行してください。

• KILL では 1 度に 1 つのファイルしか削除できません。

• KILL はすべてのディスクファイル(プログラム，ランダムファイル，シーケンシャルファイル)について使用できます。

注意　ドライブ番号はファイル名の前に "n : " と指定します。省略されたときはドライブ 1 が指定されたとみなされます。したがって複数のドライブの中に同じファイル名で指定されたフロッピィディスクがある場合注意が必要です。

参照　SET, ATTR$

サンプルプログラム

```
100 OPEN "test2" FOR OUTPUT AS #1
110 PRINT #1,"KILL statement test"
120 CLOSE
130 '
140 FILES:PRINT
150 KILL "test2"
160 FILES
```

```
run
linput.#   1    input#.    1    lof   .   1    print#.   1    dsko$ .   1
eof   .    1    dskf1 .    1    kill  .   1    get   .   1    field .   1
tdata      1    stest  dat 1    close .   1    test2     1

linput.#   1    input#.    1    lof   .   1    print#.   1    dsko$ .   1
eof   .    1    dskf1 .    1    kill  .   1    get   .   1    field .   1
tdata      1    stest  dat 1    close .   1
Ok
```

LEFT$ [N80] [N] 文字関数

機能 **文字列の左側から指定された長さの文字列を与えます。**

書式 **LEFT$(〈文字列〉, 〈数式〉)**

文例 **B$=LEFT$(A$, 3)**

解説

• LEFT$ は〈文字列〉の左側から〈数式〉で指定された長さの文字列を取り出す関数です。

• 〈数式〉は 0 ～255の範囲になければなりません。

• 〈数式〉が，〈文字列〉の総文字数以上の場合は，〈文字列〉全体をまた，〈数式〉が 0 ならばヌルストリングを与えます。

サンプルプログラム

```
100 '-- ﾓｼﾞﾚﾂ(10) ﾉ ｶｲﾃﾝ (<--) --
110 INPUT A$
120 FOR I=1 TO 10
130   PRINT A$
140   B$=LEFT$(A$,1)
150   A$=MID$(A$,2)+B$
160 NEXT I
170 END
run
? 0123456789
0123456789
1234567890
2345678901
3456789012
4567890123
5678901234
6789012345
7890123456
8901234567
9012345678
Ok
```

LEN [N80] [N] 文字関数

(機能) 文字列の長さを与えます。

(書式) LEN(<文字列>)

(文例) PRINT LEN(A$)

(解説) • LEN 関数は文字列のほかに空白や文字として画面に表示されない文字も数えます。文字として画面に表示されない文字とはキャラクタコードが 1H から 1FH までのコントロールコードと呼ばれるものをいいます。

(サンプルプログラム)

```
100 INPUT A$
110 L=LEN(A$)
120 GOSUB 170
130 PRINT "* ";A$;" *"
140 GOSUB 170
150 END
160 'flame
170 FOR I=1 TO L+4
180 PRINT "*";
190 NEXT I
200 PRINT
210 RETURN

run
? NEC PC8001MKⅡ  N80-BASIC
*****************************
* NEC PC8001MKⅡ  N80-BASIC *
*****************************
Ok
```

LET　N80 N　一般ステートメント

機能　変数に値を代入します。

書式　〔LET〕〈変数名〉＝〈式〉

文例　LET A＝B

解説　• キーワード LET は省略してもかまいません。

• 〈式〉の内容は，数値，文字列のいかなる型であってもかまいません。ただし，右辺が数値式のとき左辺は数値式でなければなりません。また，右辺が文字式のとき左辺は文字式でなければなりません。

サンプルプログラム

```
100 LET A=123
110 LET B=.21
120 C=A+B
130 PRINT "A   =";A
140 PRINT "B   =";B
150 PRINT "A+B =";C
160 END
run
A   = 123
B   = .21
A+B = 123.21
Ok
```

LINE [N80] [N] グラフィックステートメント

機能

1） テキスト画面を行単位に機能設定します。

2） 画面にキャラクタで，直線や長方形をかきます。

3） 低解像度グラフィックスで直線や長方形をかきます。

書式

1） LINE 〈行指定〉，〈ファンクションコード〉

2） LINE (X_1, Y_1)－(X_2, Y_2),〈文字列〉〔,〈ファンクションコード〉〔, {B | BF}〕〕

3） LINE (Lx_1, Ly_2)－(Lx_2, Ly_2),〈機能〉〔,〈ファンクションコード〉〔, {B | BF}〕〕

文例

1） LINE 7, 1

2） LINE(0, 0)－(39, 19), "♥", B

3） LINE(0, 0)－(100, 50), PSET, 4, BF

解説

1）• カラーモードの画面において，行単位に機能を設定します。

• 〈行指定〉は0から(1画面あたりの行数)－1までの値をとり，〈ファンクションコード〉は次の様になっています。

0	または	4	—	ノーマル
1	または	5	—	ブリンク
2	または	6	—	リバース
3	または	7	—	リバースブリンク

文例では，画面の8行目（即ち第7行）がブリンクします。なお，この命令は白黒モードでは "Syntax error" になり実行できません。

2）• 画面にキャラクタを使って，線を引いたり箱を書いたりします。

・〈文字列〉の最初の文字列を使ってキャラクタ座標の２点 (X_1, Y_1), (X_2, Y_2) を結ぶ直線を引きます。X_1, X_2 は水平位置を表し，０から(１行あたりの桁数)－１の値，Y_1, Y_2 は垂直位置を表し，０から(１画面あたりの行数)－１の値をとります。〈ファンクションコード〉は，COLORに用いられるものと同じです。

・また最後にＢを付けると (X_1, Y_1), (X_2, Y_2) の２点を角とする四角形の枠を描き，BF を付けるとその枠の内側もキャラクタで埋めます。

3)・低解像度グラフィックにより線や箱を書きます。

・基本的な使い方は2)の場合と同じですが，キャラクタではなくピクセルで線を引きます。したがって与える座標はキャラクタ座標ではなくグラフィック座標です。

・〈機能〉は，PSET または PRESET のどちらかを指定します。PSET で線を書き PRESET で線を消します。

・〈ファンクションコード〉は COLOR で指定されるものと同じです。

サンプルプログラム

```
100 CONSOLE ,,,1
110 LOCATE 0,0
120 PRINT "abcdefghijklmnopurstuvwxyz"
130 LINE 0,1
```

```
100 CONSOLE ,,,1
110 LINE(0,0)-(10,10),"♥",2
120 LINE(1,1)-(9,9),"◆",5,BF
```

```
100 CONSOLE ,,,1
110 LINE(0,0)-(100,50),PSET,5
120 LINE(10,10)-(90,40),PSET,7,B
```

LINE INPUT　[N80] [N]　入出力ステートメント

機　能　**1行全体(255文字以内)を区切ることなく文字変数に入力します。**

書　式　**LINE INPUT〔<"プロンプト文"> ; 〕<文字変数名>**

文　例　**LINE INPUT "NAME" ; NA$**

解　説

• <"プロンプト文">は入力データを受ける前に画面に表示される文章です。

• 入力を促す疑問符(？)は表示されません。

• [RETURN]キーの入力までにキーボードから入力された文字列データが文字変数に代入されます。また，文字列データを入力せず [RETURN] キーだけを押した場合文字変数はヌルストリングが入力されたとみなされます。

• LINE INPUT の実行は[CTRL]＋[C]または[STOP] によって中断することができます。その場合，コマンドレベルに戻り，"Ok" と表示されます。このとき CONT コマンドを入力すると LINE INPUT の初めから実行が再開されます。

サンプルプログラム

```
100 INPUT "INPUT : ";A$
110 LINE INPUT "LINE INPUT : ?";B$
120 PRINT:PRINT "INPUT : ",A$
130 PRINT "LINE INPUT : ",B$

run
INPUT : ? abcd,ABCD,1234
?Extra ignored
LINE INPUT : ?abcd,ABCD,1234,"Hellow"

INPUT :       abcd
LINE INPUT :  abcd,ABCD,1234,"Hellow"
Ok
```

LINE INPUT# [ディスク] [N88] [N] 入出力ステートメント

機能 **1行全体(255文字以内)を区切ることなくシーケンシャルファイルより文字変数に代入します。**

書式 **LINE INPUT#〈ファイル番号〉, 〈文字変数〉**

文例 **LINE INPUT#1, A$**

解説

• 〈ファイル番号〉はOPENによって，そのファイルを入力としてオープンしたときに指定した番号です。

• 〈文字変数〉は，行全体が代入される文字変数名です。

• LINE INPUT # 文はシーケンシャルファイルより，CR コードまでの，すべての文字を区切ることなく読み込みます。読み込める文字数は255文字までです。

• LINE INPUT #はデータファイルのそれぞれの行がいくつかの欄に分割されているときや，アスキーセーブされたプログラムをデータとして読み込むときなどに有効です。

サンプルプログラム

```
100 OPEN "tdata" FOR INPUT AS #1
110 LINE INPUT #1,A$
120 PRINT A$
130 CLOSE

run
1234567890,abcdefghijklmn,"opqrstuvwxyz"
Ok
```

LIST/LLIST　N80 N　一般コマンド

機　能　**メモリ内にあるプログラムの全部または一部を画面あるいはプリンタにリストアウトします。**

書　式　1)　**LIST**　〔〈行番号〉〕〔－〈行番号〉〕

2)　**LLIST**　〔〈行番号〉〕〔－〈行番号〉〕

文　例　**LIST 100−200**

解　説　• プログラムの内容を１)は画面に，２)はプリンタに出力します。LIST コマンドは実行し終わると，コマンドレベルに戻ります。

• 〈行番号〉が省略された場合は，最も若い行番号のプログラムよりリストアウトが始まります。

• 最初の〈行番号〉のみが指定された場合には，その指定された行だけがリストアウトされます。最初の〈行番号〉とそれに続くハイフンまでが指定された場合は，その行番号以降の行がすべてリストアウトされます。両方の行番号がハイフンでつながれて指定された場合には，その範囲のすべての行がリストアウトされます。

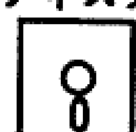
サンプルプログラム

```
110 N=1
120 'start
130 PRINT N
140 N=N+1
150 IF N<10 THEN 130
160 END

list 110-140
110 N=1
120 'start
130 PRINT N
140 N=N+1
Ok
list
140 N=N+1
Ok
list -130

110 N=1
120 'start
130 PRINT N
Ok
```

```
list 140-
140 N=N+1
150 IF N<10 THEN 130
160 END
Ok
```

LOAD    入出力コマンド

機 能　**プログラムをフロッピィディスクからメモリにロードします。**

書 式　**LOAD<ファイル名>〔,R〕**

文 例　**LOAD "2 : date"**

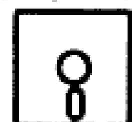
解 説　・<ファイル名>で指定されたプログラムファイルをメモリ上にロードします。ロードを実行すると，以前メモリ上にあったプログラムは自動的に NEW され，すべての開いているファイルを閉じ，変数はその値が失なわれますが，R オプションをつけると，ファイルは開いたままで，プログラムをロード後，ただちに実行を開始します。

・LOAD は，指定されたファイルを見つけて，実際のプログラムのロードを開始するまでは，メモリ中のプログラムを保存します。

参 照　SAVE, MERGE, NEW

サンプルプログラム

```
load"2:CAP-X1"
Ok
load"jusho"
Ok
```

LOC  Ⓝ80 Ⓝ 入出力関数

機能 フロッピィディスクファイル中での論理的な現在位置を与えます。

書式 LOC(〈ファイル番号〉)

文例 LAST＝LOC(1)

解説 •〈ファイル番号〉で指定されたファイルがランダムファイルであった場合には，LOC 関数は，そのランダムファイルに対して最後に読み書き(GET/PUT)されたレコード番号を返します。

•〈ファイル番号〉で指定されたファイルが，シーケンシャルファイルであった場合には，そのファイルがオープンされてから，読み書きされたレコード数を返します。

サンプルプログラム

```
100 OPEN "1:data6" AS #1
110 FIELD #1,20 AS A$,8 AS B$
120 INPUT "ﾅﾏｴ";NA$:INPUT "ｷｭｳﾖ";KY#
130 LSET A$=NA$:RSET B$=MKD$(KY#):PUT #1
140 PRINT NA$;" ｻﾝ ﾉ data ﾊ";LOC(1);"ﾊﾞﾝ ﾃﾞｽ｡"
150 INPUT "continue?(y/n)";C$
160 IF C$="y" THEN PRINT:GOTO 120
170 IF C$<>"n" THEN 150 ELSE END

run
ﾅﾏｴ? ｷﾑﾗ
ｷｭｳﾖ? 200000
ｷﾑﾗ ｻﾝ ﾉ data ﾊ 1 ﾊﾞﾝ ﾃﾞｽ｡
continue?(y/n)? y

ﾅﾏｴ? ｶﾄｳ
ｷｭｳﾖ? 170000
ｶﾄｳ ｻﾝ ﾉ data ﾊ 2 ﾊﾞﾝ ﾃﾞｽ｡
continue?(y/n)? n
Ok
```

LOCATE　N88　N　画面制御ステートメント

機能　テキスト画面のカーソル位置を指定します。

書式　**LOCATE〈X〉，〈Y〉〔，〈カーソルスイッチ〉〕**

文例　**LOCATE 10, 10**

解説

・カーソルをテキスト画面のキャラクタ座標〈X〉，〈Y〉の位置へ移動します。

・〈X〉は水平座標を表し，〈Y〉は垂直座標を表します。

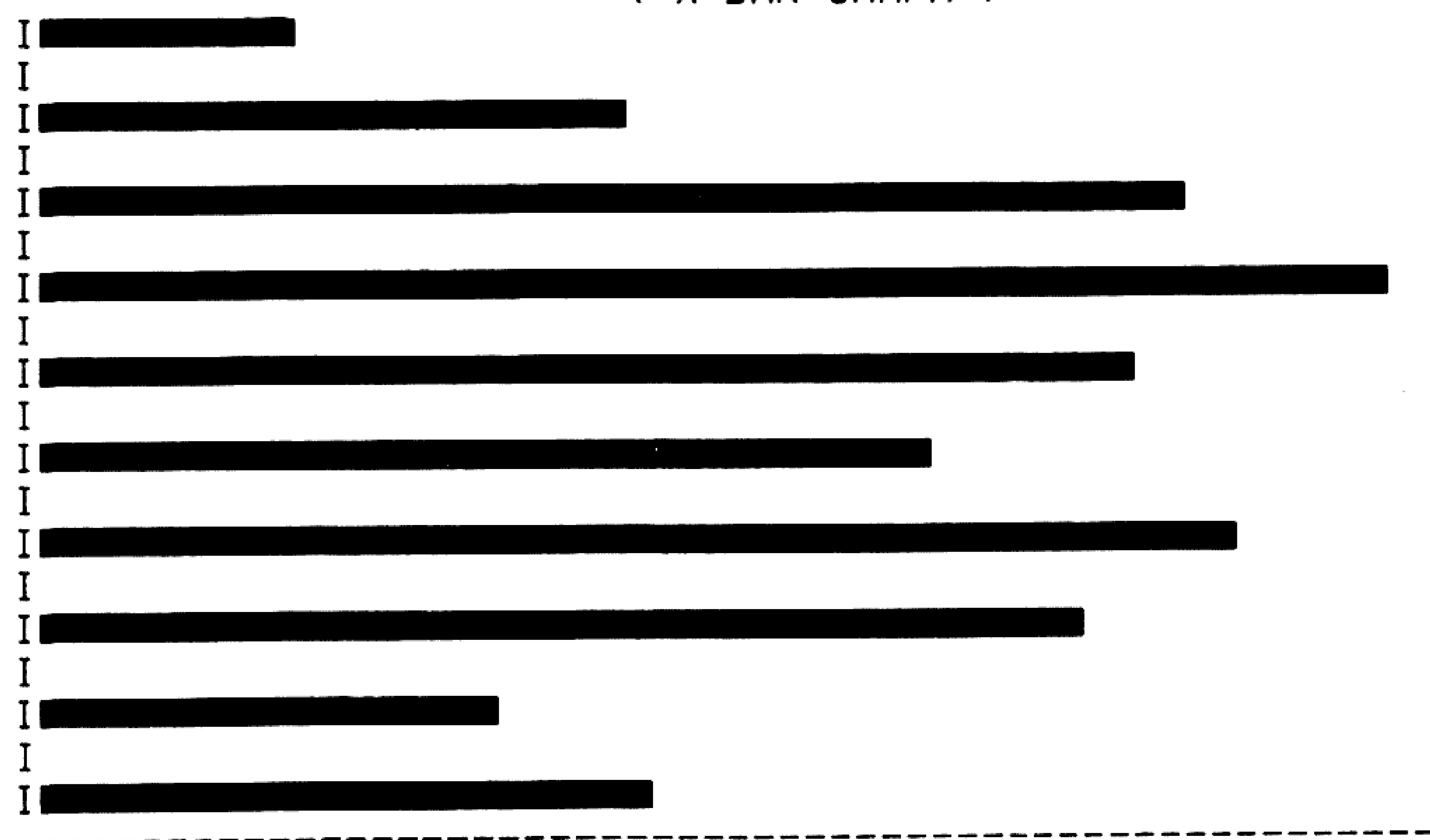
・〈X〉，〈Y〉はテキスト画面の左上を０，０とするキャラクタ座標により指定します。また〈X〉，〈Y〉の範囲はWIDTHで指定された〈桁数〉と〈行数〉によって決まり，次のようになります。

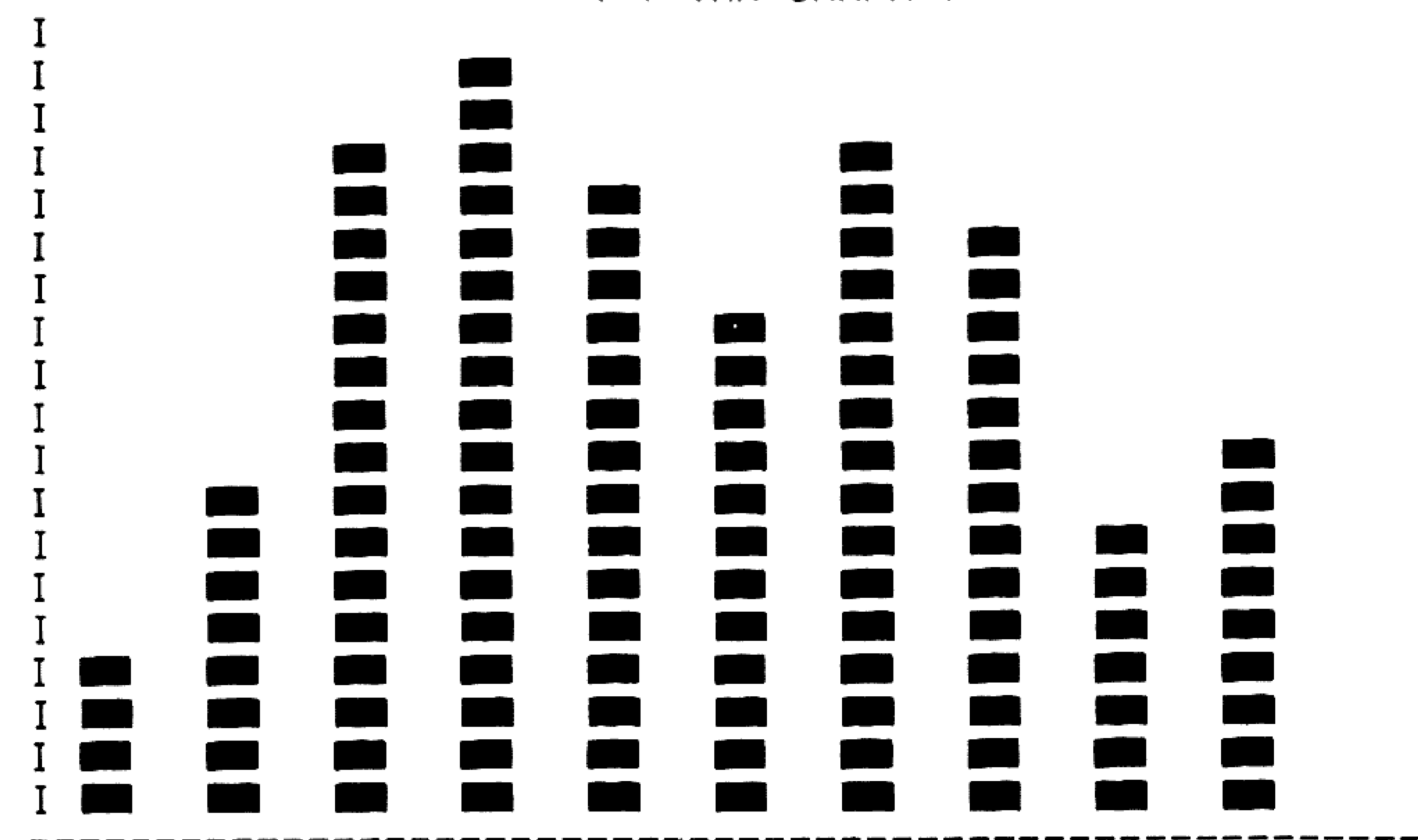
〈X〉＝０　～（〈桁数〉－１）
〈Y〉＝０　～（〈行数〉－１）

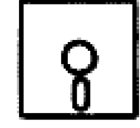
・〈カーソルスイッチ〉はカーソルの状態を決めるスイッチで，１で表示され０では表示されなくなります。

サンプルプログラム

```
100 '-- character pattern --
110 CONSOLE 0,25,0,1:WIDTH 80,25:PRINT CHR$(12)
120 DATA 10,23,45,53,43,35,47,41,18,24
130 FOR I=1 TO 10:READ DAT(I):NEXT I
140 GOSUB 180:GOSUB 240
150 A$=INPUT$(1):PRINT CHR$(12)
160 GOSUB 180:GOSUB 290:LOCATE 0,23:END
170 '-- graph draw --
180 X=3
190 FOR Y=1 TO 19
200   LOCATE X,Y:PRINT "I"
210 NEXT Y
220 LOCATE 3,20:PRINT STRING$(56,"-"):RETURN
230 'x bar
240 LOCATE 25,0:PRINT "< X BAR GRAPH >"
250 FOR I=1 TO 10
260   LOCATE 4,I*2-1:PRINT STRING$(DAT(I),CHR$(&H87))
270 NEXT I:RETURN
280 'y bar
290 LOCATE 25,0:PRINT "< Y VAR GRAPH >"
300 FOR I=1 TO 10
310   FOR J=19 TO 19-DAT(I)/3 STEP -1
320     LOCATE I*5,J:PRINT CHR$(&H87)
330     LOCATE I*5+1,J:PRINT CHR$(&H87)
340   NEXT J
350 NEXT I:RETURN
```


< X BAR GRAPH >


< Y VAR GRAPH >

LOF ディスク   入出力関数

(機　能) **ファイルの大きさを与えます。**

(書　式) **LOF(〈ファイル番号〉)**

(文　例) **MR=LOF(2)**

(解　説) • LOF 関数は〈ファイル番号〉により指定されたディスクファイルの大きさをセクタ数で返します。そのファイルがランダムファイルであればそのファイルをオープンしたときの最大のレコード番号に相当します。

• GET で LOF 関数の返す値よりも大きなレコードナンバーを指定すると "Input past end" エラーが起ります。

(サンプルプログラム)

```
100 '-- LOF of sequntial &random file --
110 OPEN "2:sdata1" FOR OUTPUT AS #1
120 PRINT #1,123:CLOSE
130 OPEN "2:sdata1" FOR INPUT AS #1
140 OPEN "2:rdata1" AS #2
150 FIELD #2,128 AS A$,128 AS B$
160 PRINT "sdata1 ﾉ ｾｸﾀ ｽｳﾊ";LOF(1);"ﾃﾞｽ｡"
170 PRINT "rdata1 ﾉ ｾｸﾀ ｽｳﾊ";LOF(2);"ﾃﾞｽ｡"
180 CLOSE

run
sdata1 ﾉ ｾｸﾀ ｽｳﾊ 1 ﾃﾞｽ｡
rdata1 ﾉ ｾｸﾀ ｽｳﾊ 0 ﾃﾞｽ｡
Ok
```

LOG　[N80] [N]　数値関数

機能　自然対数を与えます。

書式　LOG(〈数式〉)

文例　A=LOG(35/9)

解説

- 〈数式〉によって与えられた値の自然対数 (e を底とした対数)を返します。
- LOG 関数の演算は単精度で行われます。

サンプルプログラム

```
100 '-- ｼﾞｮｳﾖｳ ﾀｲｽｳ ｦ ﾓﾄﾒﾙ --
110 F$="log10(##.#####^^^^)=##.#####^^^^"
120 INPUT L
130 A=LOG(10)
140 B=LOG(L)
150 PRINT USING F$;L,B/A
160 END

run
? 100
log10( 1.00000E+02)= 2.00000E+00
Ok
run
? .1245
log10( 1.24500E-01)=-9.04831E-01
Ok
run
? 876.2
log10( 8.76200E+02)= 2.94260E+00
Ok
```

LPOS　N80 N　入出力関数

機能　現在のプリンタのヘッド位置を与えます。

書式　LPOS(〈式〉)

文例　HE=LPOS(0)

解説　• 〈式〉の値はダミーであり意味を持ちません。

サンプルプログラム

```
100 FOR I=0 TO 20
110   IF LPOS(0)>=40 THEN LPRINT
120   LPRINT SQR(I),
130 NEXT I
140 END
run
 0             1             1.41421
 1.73205       2             2.23607
 2.44949       2.64575       2.82843
 3             3.16228       3.31663
 3.4641        3.60555       3.74166
 3.87298       4             4.12311
 4.24264       4.3589        4.47214
Ok
```

LSET/RSET ディスク N80 N 入出力ステートメント

機能 ファイルバッファへデータを書き込みます。

書式 LSET<文字型変数>＝<文字列>

RSET<文字型変数>＝<文字列>

文例 LSET A$＝"N-BASIC"

RSET B$＝"N80-BASIC"

解説 • Left SET, Right SET の意味でランダムファイルのレコードの，FIELD で定義した文字変数に該当する領域にデータを格納します。

• 格納するデータは文字型データでなければならず，数値データはMKI$, MKS$, MKD$のいずれかにより文字型データに変換しなければなりません。

• <文字列>の文字数が FIELD で割り当てられた長さより短い場合，LSET では左詰めでフィールドを満たします。また，<文字列>がフィールドより長い場合は，LSET, RSET とも右側の部分が失われます。

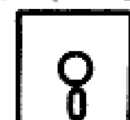
• <文字型変数>は，FIELD であらかじめ定義されていなければなりません。定義されていない変数を用いるとエラーが発生します。

参照 FIELD, PUT

サンプルプログラム

```
100 OPEN "1:rdata" AS #1
110 FIELD 1,20 AS A$,10 AS B$
120 LINE INPUT "ﾋﾝﾒｲ ?";NA$
130 INPUT "ﾀﾝｶ ";PR%
140 LSET A$=NA$
150 RSET B$=MKI$(PR%)
160 PUT #1,1
170 CLOSE
Ok

run
ﾋﾝﾒｲ ?ﾒﾝﾀｲｺ
ﾀﾝｶ ? 300
Ok
```

MERGE ディスク    入出力コマンド

機能 **メモリ上にあるプログラムとフロッピィディスク上のプログラムファイルを混合します。**

書式 **MERGE<ファイル名>**

文例 **MERGE "2 : TEST1"**

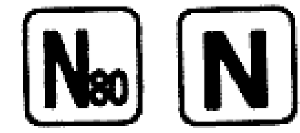
解説 • メモリ上のプログラムと<ファイル名>により指定したプログラムファイルを1つにして，メモリ上に置きます。ここで，指定されたファイルはアスキーセーブされていなければなりません。そうでない場合には，エラーになります。

• ファイル中のプログラムと，メモリ上のプログラムに同一行番号があった場合には，メモリ上の行はファイル中の行で置き換えられます。

参照 SAVE

サンプルプログラム

```
merge"a-5"
Ok
merge"2:CAP-X1"
Ok
```

(1)MID$  文字ステートメント

機能 **文字列の一部を指定された文字列で置き換えます。**

書式 **MID$(〈文字列1〉, 〈式1〉〔, 〈式2〉〕)=〈文字列2〉**

文例 **MID$(A$, 3)="ABC"**

解説 • 〈文字列 1 〉の〈式 1 〉番目の文字から〈式 2 〉個の文字を，〈文字列 2 〉の最初から〈式 2 〉個の文字列で置き換えます。

• 〈式 2 〉が省略された場合，または〈文字列 2 〉の文字数より多く指定した場合は，〈文字列 2 〉のすべての文字と置き換わります。

• 〈式 1 〉+〈式 2 〉− 1 の値が〈文字列 1 〉の文字数より大きくなり，指定した文字列があふれてしまう場合は，(〈文字列 1 〉の文字数)−〈式 1 〉+ 1 の値を〈式 2 〉とし，〈文字列 2 〉の残りの部分は無視します。

• 〈式 1 〉の値は〈文字列 1 〉の文字数より大きくてはいけません。また，0 以下であってもいけません。したがって〈文字列 1 〉にヌルストリングを指定することはできません

サンプルプログラム

```
100 DATA My name is !!!!!!!!
110 DATA Arai,Suzuki,Komatsu,Watanabe
120 READ A$
130 FOR I=1 TO 4
140   READ NA$:MID$(A$,12)=NA$
150   PRINT A$
160 NEXT
Ok

run
My name is Arai!!!!
My name is Suzuki!!
My name is Komatsu!
My name is Watanabe
Ok
```

(2)MID$　N80 N　文字関数

機能　**文字列の指定された位置から任意の長さの文字列を与えます。**

書式　**MID$(〈文字列〉, 〈式1〉〔, 〈式2〉〕)**

文例　**B$=MID$(A$, 2, 3)**

解説

• 〈式1〉は，〈文字列〉の左側から数え，文字列を取り出す最初の位置を示します。また〈式1〉は1～255の範囲になければなりません。

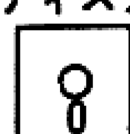
• 〈式2〉は，〈式1〉で示された位置から取り出す文字数を指定します。〈式2〉は0～255の範囲です。

• 〈式2〉を省略したとき，また，〈式2〉が〈式1〉番目から右の文字数(つまり，残りの文字数)より大きくなったとき，〈式1〉番目より右の文字列全部が結果となります。

• 〈式2〉が0のとき，または，〈文字列〉全体の文字数が，〈式1〉より小さい場合，MID関数はヌルストリングを与えます。

サンプルプログラム

```
100 WIDTH 40,25:CMD CLS 1
110 A$="PC-8001MKⅡ"+STRING$(10," ")
120 I=1
130   LOCATE 10,10
140   GOSUB 220
150   PRINT A$
160   B$=LEFT$(A$,1)
170   A$=MID$(A$,2)+B$
180   I=I+1
190   IF I>30000 THEN I=0
200 GOTO 130
210 'wait routine
220 FOR I=1 TO 100:NEXT
230 RETURN
```

MKI$/MKS$/MKD$    文字関数

(機　能) **各数値を内部表現に対応した文字コードを与えます。**

(書　式) 1) **MKI$(〈整数表記〉)**

2) **MKS$(〈単精度表記〉)**

3) **MKD$(〈倍精度表記〉)**

(文　例) 1) **A$=MKI$(128)**

2) **LSET B$=MKS$(1.23)**

3) **RSET C$=MKD$(3.14159265389#)**

(解　説) ・これらの関数は，数値をランダムファイル・バッファに対してLSET/RSET命令で書き込む際に使用します。数値から文字への変換は数値が持つ内部表現(2進数表現)の値をそのままそれに対応する文字コードにすることによって行われます。この逆の動作をする関数としてはCVI/CVS/CVDが用意されています。

1) 整数値を2文字(2バイト)の文字列に変換します。

2) 単精度数値を4文字(4バイト)の文字列に変換します。

3) 倍精度数値を8文字(8バイト)の文字列に変換します。

(注　意) STR$関数とは変換の仕方が異なります。

(参　照) CVI/CVS/CVD, STR$, 付録A, PC8001MKII ユーザーズマニュアル

(サンプルプログラム)

```
100 '-- random data put --
110 OPEN ":rdata" AS #1
120 FIELD #1,2 AS D1$,4 AS D2$,8 AS D3$
130 A%=123
140 B=8765.43
150 C#=3.14159265358979#
160 LSET D1$=MKI$(A%)
170 LSET D2$=MKS$(B)
```

```
180 LSET D3$=MKD$(C#)
190 PUT #1,1
200 '--random data get --
210 GET #1,1
220 A%=CVI(D1$)
230 B=CVS(D2$)
240 C#=CVD(D3$)
250 PRINT "A =";A%
260 PRINT "B =";B
270 PRINT "C =";C#
280 CLOSE
290 END

run
A = 123
B = 8765.43
C = 3.14159265358979
Ok
```

MON　[N80] [N]　一般コマンド

機能　**内蔵の機械語モニタに制御を移します。**

書式　**MON**

文例　**MON**

解説

• BASIC モードから内蔵の機械語モニタに制御を移すためのコマンドです。

• モニタがコマンドレベルに入ると画面の左端に"＊"が表示されます。

• モニタには次のようなコマンドがあります。詳しくは PC-8001MKⅡ ユーザーズ マニュアルを参照してください。

コマンド	内　容
D	メモリの内容をディスプレイに表示します。
G	機械語プログラムの実行をします。
L	カセットテープからメモリへロードします。
LV	カセットテープの内容とメモリの内容を比較します。
S	メモリの内容を変更します。
TM	メモリのテストを行います。
W	メモリの内容をカセットにセーブします。
CTRL ＋ B	BASIC に戻ります。

• BASIC のコマンドレベルに戻るには CTRL ＋ B を入力してください。

参照　PC-8001MK II ユーザーズマニュアル

サンプルプログラム

```
mon
*D0,2F
0000 F3 31 FF FF C3 3B 00 00 C3 6A 00 C3 57 17 AB F0
0010 C3 59 42 C3 6A 00 DA 0C C3 A6 40 F3 0B C3 7E 50
0020 C3 DA F1 C3 9C 27 88 0C C3 DD F1 C3 60 0D 46 0C
*
Ok
```

MOTOR　N80 N　入出力コマンド

機能　カセットテープレコーダのモータの ON,OFF を制御します。

書式　**MOTOR〔〈スイッチ〉〕**

文例　**MOTOR 1**

解説

• 〈スイッチ〉を 0 にすればモータは OFF に，0 以外の値にすれば ON の状態になります。

• 〈スイッチ〉を省略した場合，モータが OFF の状態なら ON に，ON の状態なら OFF になります。

参照　PC-8001MKII ユーザーズマニュアル

サンプルプログラム

```
100 MOTOR 1
110 PRINT "Rewind and set then tape."
120 INPUT "Are you ready (y/n)";A$
130 IF LEFT$(A$,1)<>"y" THEN END
140 PRINT
150 MOTOR 0
160 PRINT "Push PLAY button."
170 INPUT "Are you ready (y/n)";A$
180 IF LEFT$(A$,1)<>"y" THEN END
190 MOTOR
200 END

run
Rewind and set then tape.
Are you ready (y/n)? y

Push PLAY button.
Are you ready (y/n)? y
Ok
```

MOUNT  入出力コマンド

(機　能) **ディスクのファイル配置表(FAT)を読んで以後ディスク入出力を可能にします。**

(書　式) **MOUNT〔<ドライブ番号>〔,<ドライブ番号>……〕〕**

(文　例) **MOUNT**

(解　説) • フロッピィディスクとの入出力 (LOAD, OPEN, PRINT #) などを行う前に MOUNT を実行しなければなりません。

• <ドライブ番号>を省略した場合，現在接続されているすべてのドライブが MOUNT されます。

• MOUNT を実行すると BASIC はディスケットからファイルの配置表を読み込み，誤りを調べます。誤りがなければ，そのフロッピィディスクはマウントされます。もし誤りがあった場合には，BASIC はフロッピィディスクをマウントするために予備の配置表を読み込もうとします。その場合，次のメッセージが表示されます。“X copies of allocation bad on drive Y” X は 1 または 2，Y はドライブ番号です。この警告が起きた場合は，ファイルの配置表に何らかのトラブルが発生している訳ですから，新しくフロッピィディスクのコピーを作ることをお勧めします。もし予備の配置表のコピーが悪ければ，致命的なエラーである “Bad allocation table” となり，そのフロッピィディスクはマウントされません。

• フロッピィディスクがマウントされている間に，BASIC は時折そのフロッピィディスクに配置表を書き込みますが，そのドライブにリード・アフターライトのアトリビュートがセットされていなければ，エラーチェックを行いません。(SET 参照。)

• N80 DISK-BASIC ではオート MOUNT, REMOVE を採用して

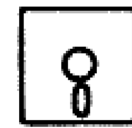
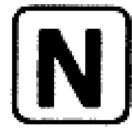
いるため，これらの操作はすべてBASICが管理します。

• N80DISK-BASICでMOUNTは実行されません。エラーは起らず，プログラムの実行に影響を及ぼすこともありません。

参照　REMOVE, SET

サンプルプログラム

```
mount 1
Ok
mount 2
Ok
```

NAME    一般コマンド

機能 **フロッピィディスク上のファイルの名前を変えます。**

書式 **NAME〈旧ファイル名〉AS〈新ファイル名〉**

文例 **NAME "2 : sample" AS "2 : SAMPLE.2"**

解説

- 〈旧ファイル名〉で表わされているフロッピィディスク上のファイルを〈新ファイル名〉に変更します。
- ドライブの指定は〈旧ファイル名〉,〈新ファイル名〉中の先頭にドライブ番号をコロン(:)で区切ることにより行われます。
- ドライブ番号を省略した場合,ドライブ1が指定されます。
- NAMEによりファイル名の変更を行うとき,そのファイルはクローズされた状態でなければなりません。

参照 KILL

サンプルプログラム

```
100 OPEN "oldtest" FOR OUTPUT AS#1
110 FOR J=1 TO 10
120   PRINT #1,J
130 NEXT J:CLOSE
140 FILES:PRINT
150 NAME "oldtest" AS "newtest"
160 FILES

run
Thank     1     you!      1     Darlin g!  1     oldtes t    1


Thank     1     you!      1     Darlin g!  1     newtes t    1

Ok
```

NEW [N80] [N] 一般コマンド

機能 メモリ上にあるプログラムを抹消し，すべての変数をクリアします。

書式 NEW

文例 NEW

解説

• NEW は，コマンドレベルにあるとき新しいプログラムを入力する前に実行します。NEW の実行が終わると，いつもコマンドレベルに戻ります。

• NEW は，オープンされているファイルがある場合，それを自動的にすべてクローズします。

サンプルプログラム

```
10 'NEW sample
20 'NEW initializes the program.
30 PRINT "Don't use NEW in a program like this!"
40 PRINT "Program is deleted."
50 NEW

run
Don't use NEW in a program like this!
Program is deleted.
Ok
list
Ok
```

OCT$ N80 N 文字関数

機能 10進数を8進数の文字列に変換します。

書式 OCT$(〈数式〉)

文例 PRINT OCT$(A)

解説

• 〈数式〉の値を8進数に変換して，その文字列を与える関数です。

• 〈数式〉の値は，−32768〜32767(または0〜65535)までです。また，〈数式〉に小数点が含まれる場合，小数点以下を切捨てます。

サンプルプログラム

```
100 '** decimal --> octal -->hexadecimal **
110 FOR I=0 TO 16
120   DE$=RIGHT$("  "+STR$(I),3)
130   OC$=RIGHT$("  "+OCT$(I),3)
140   HE$=RIGHT$("  "+HEX$(I),3)
150   PRINT "DECIMAL:";DE$,"OCTAL:";OC$,"HEX:";HE$
160 NEXT I

run
DECIMAL:  0    OCTAL:  0      HEX:  0
DECIMAL:  1    OCTAL:  1      HEX:  1
DECIMAL:  2    OCTAL:  2      HEX:  2
DECIMAL:  3    OCTAL:  3      HEX:  3
DECIMAL:  4    OCTAL:  4      HEX:  4
DECIMAL:  5    OCTAL:  5      HEX:  5
DECIMAL:  6    OCTAL:  6      HEX:  6
DECIMAL:  7    OCTAL:  7      HEX:  7
DECIMAL:  8    OCTAL: 10      HEX:  8
DECIMAL:  9    OCTAL: 11      HEX:  9
DECIMAL: 10    OCTAL: 12      HEX:  A
DECIMAL: 11    OCTAL: 13      HEX:  B
DECIMAL: 12    OCTAL: 14      HEX:  C
DECIMAL: 13    OCTAL: 15      HEX:  D
DECIMAL: 14    OCTAL: 16      HEX:  E
DECIMAL: 15    OCTAL: 17      HEX:  F
DECIMAL: 16    OCTAL: 20      HEX: 10
Ok
```

ON ERROR GOTO [N80] [N] 特殊ステートメント

機能 **エラートラップを可能にし，エラー処理ルーチンの開始行を定義します。**

書式 **ON ERROR GOTO〈行番号〉**

文例 **ON ERROR GOTO 1000**

解説

• エラートラップ機能が可能になると，エラーが検出されたとき指定されたエラー処理ルーチンへプログラムの制御が移ります。

• 〈行番号〉の行が存在しなければ，"Undefined Line"（未定義行）のエラーが起こります。

• エラートラップ機能を無効にするにはON ERROR GOTO 0を実行してください。

• エラー処理サブルーチンの実行中にエラーが起きた場合には，それに対するエラーメッセージが表示され，実行は停止します。エラー処理サブルーチンの中では，エラートラップは起こりません。

• エラー処理後，プログラムの実行を再開するにはRESUMEを使います。

参照 RESUME

サンプルプログラム

```
100 ´ X ﾉ y ｼﾞｮｳ ｦ ﾓﾄﾒﾙ
110 ON ERROR GOTO 180
120 ´start
130 INPUT "X :";X:INPUT "y :";Y
140 IF X=0 THEN ERROR 250
150 Z=X^Y:PRINT "   y":PRINT " X =";Z
160 GOTO 130
170 ´error message
180 IF ERR=250 THEN PRINT " ﾃｲｷﾞ ｻﾚﾏｾﾝ｡"
190 IF ERR=6   THEN PRINT "ｵｰﾊﾞｰ ﾌﾛｰ ﾃﾞｽ｡"
200 RESUME 130

run
X :? 2
y :? 10
  y
 X = 1024
```

```
X :? 100
y :? 100
ｵｰﾊﾞｰ ﾌﾛｰ ﾃﾞｽ｡
X :? 0
y :? -2
 ﾃｲｷﾞ ｻﾚﾏｾﾝ｡
X :?
Break in 130
Ok

a=usr(0)
Ok
```

ON…GOSUB/ON…GOTO N80 N
一般ステートメント

(機　能) 指定されたいくつかの行に分岐します。

(書　式) ON <数式> GOSUB<行番号>〔, <行番号>…〕

ON <数式> GOTO<行番号>〔, <行番号>…〕

(文　例) ON A GOSUB 100, 200, 300

ON B GOTO 400, 500, 600

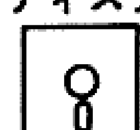
(解　説) • <数式>の値がプログラムのどの行番号の行に分岐するかを決定します。<行番号>の並びは1から始まる数に対応します。たとえば，<数式>の値が3であったなら，行番号のリストの3番目の行が分岐点となります。

• <数式>の値が負になった場合には，"Illegal function call" エラーが起こりますが，値が0または行番号のリストの個数より大きくなった場合には,次の行へプログラムの制御が移り，エラーは起こりません。

• ON…GOSUBでは,リストの各行番号はサブルーチンの先頭の行番号でなければなりません。

(サンプルプログラム)

```
100 'entry
110 INPUT "N =";N
120 SG=SGN(N)+1
130 ON SG GOTO 160,180
140 PRINT "MINUS!":GOTO 110
150 'zero
160 PRINT "ZERO!":GOTO 110
170 'plus
180 PRINT "PLUS!":GOTO 110
Ok
run
N =? 0
ZERO!
N =? 987
PLUS!
N =? -9897
MINUS!
N =?
Break in 110
Ok
```

OPEN    入出力ステートメント

(機　能) **フロッピィディスク上のファイルをオープンします。**

(書　式) **OPEN〈ファイル名〉〔FOR〈モード〉〕AS〔#〕〈ファイル番号〉**

(文　例) **OPEN "2 : TEST" FOR INPUT AS #1**

(解　説) • 〈ファイル名〉で指定されたフロッピィディスク上のファイルを，指定された〈ファイル番号〉でオープンします。以後，OPEN されたファイルへの入出力は，〈ファイル番号〉を指示することにより行います。

• 〈モード〉は，ファイルのアクセス方法を指定します。モードには次の 4 種類があります。

INPUT …………既存のシーケンシャルファイルから入力を行うことを指示します。
OUTPUT ………新しくシーケンシャルファイルを作り出力を行うことを指示します。
APPEND ………既存のシーケンシャルファイルの終りから追加を行うことを指示します。
省略したとき…FOR〈モード〉が省略されると，ランダムアクセスを行うことを指示します。

• INPUT, APPEND モードでは指定されたファイルが存在しないと，"File not found" のエラーとなります。OUTPUT モードでは常に指定された名前のファイルを新しく作り，同一名のファイルがあった場合，そのファイルは削除されます。ランダムアクセスでファイルが存在しない場合には新たにファイルが作られます。

• 〈ファイル番号〉は，1～15までの値を用いることができますが，(N80)DISK-BASIC を使い始めるときに "How many files(0–15)?" で指定したファイルの数を越えてはいけません。

• OPEN は，以後の入出力の際に用いるバッファ領域を確保し，ファイルが閉じられるまでの間，そのバッファは，指定したファイルへの入出力操作専用に使われます。このバッファはファイル番号と同一の番号で参照されます。

注意

カセットテープ上のファイルはオープンする必要がありません。

参照

CLOSE, VARPTR, FIELD

サンプルプログラム

```
100 '-- ﾃﾞｰﾀ ﾃﾝｿｳ ﾌﾟﾛｸﾞﾗﾑ --
110 '   DSK1: --> DSK2
120 '
130 OPEN "1:data3" FOR INPUT AS#1
140 OPEN "2:newdata3" FOR OUTPUT AS#1
150 '
160 'start
170 IF EOF(1) THEN 220
180    INPUT #1,A$
190    PRINT #2,A$
200    GOTO 170
210 'exit
220 CLOSE
230 END
```

OUT [N80] [N] 特殊ステートメント

(機　能) コンピュータの出力ポートに1バイトのデータを送ります。

(書　式) **OUT<I/Oアドレス>,<数式>**

(文　例) **OUT &H30, &H3**

(解　説) • <I/O アドレス>は出力ポートの番号，そして<数式>は出力する1バイトのデータです。

• <I/O アドレス>, <数式>は共に0～255までの整数で指定します。

(参　照) INP, PC-8001MKⅡ ユーザーズマニュアル付録9 I/O マップ

(サンプルプログラム)

```
100 'out test
110 FOR I=1 TO 100
120   OUT &H40,&H2F
130   OUT &H40,&HF
140 NEXT I
150 'wait
160 FOR I=1 TO 200:NEXT I
170 'beep
180 FOR I=1 TO 100
190   BEEP 1
200   BEEP 0
210 NEXT I
```

PEEK  特殊関数

機能 メモリ上の指定された番地の内容を読み出します。

書式 PEEK(<番地>)

文例 A=PEEK(&HDA00)

解説 • <番地>によって指定されたメモリ番地の内容を与えます。

• <番地>の範囲は，0～65535(0H～FFFFH)までです。また，小数点が含まれる場合，小数点以下を切捨てた後で変換を行います。

サンプルプログラム

```
110 INPUT "start address(hex)";SA$
120 SA=VAL("&H"+SA$)
130 INPUT "end   address(hex)";EA$
140 EA=VAL("&H"+EA$)
150 CO=1
160 'start
170 PRINT RIGHT$("000"+HEX$(SA),4);": ";
180 FOR I=SA TO SA+15
190   DA=PEEK(I)
200   PRINT RIGHT$("0"+HEX$(DA),2);" ";
210 NEXT I
220 SA=SA+16:PRINT
230 IF SA<=EA THEN 170
240 END

run
start address(hex)? 8000
end   address(hex)? 8050
8000: E6 F1 E9 F1 C9 00 C9 00 7F 23 A9 23 C9 00 C9 00
8010: C9 00 C9 00 C9 00 C9 00 C9 00 C9 00 C9 00 C9 00
8020: 00 40 80 6E 00 85 20 22 73 74 61 72 74 20 61 64
8030: 64 72 65 73 73 28 68 65 78 29 22 3B 53 41 24 00
8040: 54 80 78 00 53 41 F1 FF 94 28 22 26 48 22 F3 53
8050: 41 24 29 00 73 80 82 00 85 20 22 65 6E 64 20 20
Ok
```

POINT [N80] [N] グラフィック関数

(機　能) 指定した低解像度グラフィック座標にピクセルがセットされているかいないかを与えます。

(書　式) POINT(Lx, Ly)

(文　例) CL=POINT(60, 60)

(解　説) • 指定した低解像度グラフィック座標に，ピクセルがセットされている場合には－1，リセットされていたり，キャラクタが表示されている場合には0を与えます。

(サンプルプログラム)

```
100 CMD CLS 1
110 LINE(0,50)-(159,50),PSET
120 X=50:Y=0
130 IF POINT(X,Y)=-1 THEN END
140 PSET(X,Y)
150 Y=Y+1
160 GOTO 130
```

POKE [N80] [N] 特殊ステートメント

機能 **メモリ上の指定番地へデータを書き込みます。**

書式 **POKE〈番地〉, 〈整数表記〉**

文例 **POKE &HE000, &HFF**

解説 • 指定されたメモリ上の番地に，1バイト(8ビット)のデータを書き込みます。(ただし，RAMの領域に限ります)

• 〈整数表記〉は書き込まれるデータで，0～255(0H～FFH)の値でなければなりません。もし小数点以下があると整数化してから実行されます。

• 〈番地〉は2バイトの整数で，0～65535(0H～FFFFH)の値です。ただしこの命令は，現在のメモリの内容を書き換えてしまうため，不用意に使うとBASICが使っている作業領域を壊してしまい，誤動作の原因となることもあります。使う時には，メモリマップなどで使用可能な領域かどうかを確認してください。

• POKEの逆の働きをする関数としてPEEK関数というものも用意されています(PEEK関数参照)。これらPOKEとPEEKは，効率的なデータの格納や機械語サブルーチンとの引数の受け渡しなどに役立ちます。

参照 PEEK, CLEAR, 付録B 機械語プログラムの作り方

サンプルプログラム

```
100 'POKE
110 CLEAR 300,&HDFFF
120 '-- write to memory --
130 FOR AD=&HE000 TO &HE00F
140   POKE ADD,DA:DA=DA+1
150 NEXT AD
160 '-- read from memory --
170 FOR AD=&HE000 TO &HE00F
180   PRINT RIGHT$(" 0"+HEX$(PEEK(ADD)),4);
190 NEXT AD
```

PORT　N88 N　入出力関数

機能　**指定された RS-232C 通信ポートの入出力バッファに入力されている文字数を与えます。**

書式　**PORT**(<**ポート番号**>)

文例　**PORT(1)**

解説　• <ポート番号>で指定された RS-232C 通信ポートの入力バッファに現在何文字入力されているかを知る関数です。本体内の RS-232C インタフェースを使用する場合<ポート番号>は 1 を指定します。

• RS-232C 通信ポートの入出力バッファは127文字分あり，データがそのバッファの容量を越えた場合"Communication buffer overflow" エラーが起ります。したがって，PORT 関数でバッファがある程度いっぱいになったら INPUT% などでデータの読み出しを行ってください。

サンプルプログラム

```
100 ' PORT
110 POKE &H8001,&H23:POKE &H8000,&H7F
120 OUT &HE6,4
130 OUT &HE4,&HFF
140 INIT %1,&HFA,&H37
150 IF PORT(1) THEN PRINT %1,INPUT$(PORT(1),%1);
160 GOTO 150
170 '*********************************
180 ' After execution of this program,
190 ' type "out &hE6,&h0".
200 '*********************************
```

POS　[N80] [N]　画面制御関数

機能　**テキスト画面上の現在のカーソルの水平位置を与えます。**

書式　**POS**(〈式〉)

文例　**Y＝POS(0)**

解説
- ここでの〈式〉は意味を持ちません。通常0を使います。
- 得られる値は0からそのときWIDTHで指定した〈桁数〉未満の整数となります。

参照　CSRLIN, LOCATE, WIDTH

サンプルプログラム

```
100 FOR I=32 TO 255
110   IF POS(0)=>32 THEN PRINT
120   PRINT CHR$(I);" ";
130 NEXT I

run
  ! " # $ % & ' ( ) * + , - . /
0 1 2 3 4 5 6 7 8 9 : ; < = > ?
@ A B C D E F G H I J K L M N O
P Q R S T U V W X Y Z [ ¥ ] ^ _
` a b c d e f g h i j k l m n o
p q r s t u v w x y z { ¦ } ~
▁ ▂ ▃ ▄ ▅ ▆ ▇ █ ▏ ▎ ▍ ▌ ▋ ▊ ▉ ┼
┴ ┬ ┤ ├ ▔ ─ │ ▕ ┌ ┐ └ ┘ ╭ ╮ ╰ ╯
  。 「 」 、 ・ ヲ ァ ィ ゥ ェ ォ ャ ュ ョ ッ
ー ア イ ウ エ オ カ キ ク ケ コ サ シ ス セ ソ
タ チ ツ テ ト ナ ニ ヌ ネ ノ ハ ヒ フ ヘ ホ マ
ミ ム メ モ ヤ ユ ヨ ラ リ ル レ ロ ワ ン ゛ ゜
═ ╞ ╪ ╡ ◢ ◣ ◥ ◤ ♠ ♥ ♦ ♣ ● ○ ╱ ╲
╳ 円 年 月 日 時 分 秒
Ok
```

PRESET　N80 N　グラフィックステートメント

機能　**低解像度グラフィック座標(Lx,Ly)のピクセルをリセットします。**

書式　**PRESET(Lx, Ly)〔,<ファンクションコード>〕)**

文例　**PRESET(100, 50)**

解説
- 低解像度グラフィックにおいて任意のピクセルをリセットします。
- ここでの<ファンクションコード>は意味を持たず，通常指定しません。

注意　PRESET は COLOR で指定される<グラフィックスイッチ>が ON の状態で使用してください。

参照　PSET, COLOR, CONSOLE

サンプルプログラム

```
100 CONSOLE ,,0,1
110 PRINT CHR$(12)
120 FOR I=0 TO 90
130   PSET(I,I,C)
140   C=C+1
150   C=C MOD 8
160   PRESET(I,I)
170 NEXT I
```

PRINT/LPRINT [N80] [N] 入出力ステートメント

機能 1)テキスト画面に情報を出力します。

2)プリンタに情報を出力します。

書式 1) PRINT〔<式>…〕

2) LPRINT〔<式>…〕

文例 PRINT "ABC"

解説 ・指定した式の値や文字列を1)はディスプレイ画面に2)はプリンタに出力します。

・〈式〉が省略されている場合は改行のみを行います。式の値や文字列を表示する領域は，あらかじめ各行を14文字毎に分割して定められており，式の区切り記号にコンマを使うと次の領域の始めから，またセミコロンを使うと直前にプリントしたもののすぐ後ろに続いて次の式(値や文字列)が表示されます。区切り記号として空白"△"を用いた場合もセミコロンと同様の働きをします。

・〈式〉の最後にセミコロン及びコンマを指定すると改行動作をおこさず，その行で次の PRINT による出力を続行します。

・変数と"で囲まれた文字列との区切りに限って"△"を省略することができ，このときはセミコロンと同様の働きをします。

・数値を表示した場合その後ろに必ず空白が一つ挿入されます。また，数の前に符号のための桁を確保します。単精度の数値で，指数形式でなくても 6 桁以下の桁数で精度に影響を及ぼさず表示できるものは，実数形式で表示されます。
同様に倍精度の数値は16桁以下の桁数で精度に影響を及ぼさず表示できるものは，実数形式の表示になります。

・この命令のキーワード "PRINT" の簡略形として疑問符(？)

を使用することが許されます。

• 表示する文字列の長さ，数値の長さが現在のカーソルの位置より後方にとれない場合(それを表示すると次の行にわたってしまう)改行してそれを表示します。

サンプルプログラム

```
100 FOR I=2 TO 15
110   FOR J=0 TO 15
120     PRINT CHR$(I*16+J);SPC(2);
130   NEXT J
140   PRINT
150 NEXT I
160 END
```

```
run
   !  "  #  $  %  &  '  (  )  *  +  ,  -  .  /
0  1  2  3  4  5  6  7  8  9  :  ;  <  =  >  ?
@  A  B  C  D  E  F  G  H  I  J  K  L  M  N  O
P  Q  R  S  T  U  V  W  X  Y  Z  [  ¥  ]  ^  _
`  a  b  c  d  e  f  g  h  i  j  k  l  m  n  o
p  q  r  s  t  u  v  w  x  y  z  {  ¦  }  ~
▁  ▂  ▃  ▄  ▅  ▆  ▇  █  ▏  ▎  ▍  ▌  ▋  ▊  ▉  ┼
┴  ┬  ┤  ├  ▔  ─  │  ▕  ┌  ┐  └  ┘  ╭  ╮  ╰  ╯
   。 「 」 、 ・ ヲ ァ ィ ゥ ェ ォ ャ ュ ョ ッ
ー ア イ ウ エ オ カ キ ク ケ コ サ シ ス セ ソ
タ チ ツ テ ト ナ ニ ヌ ネ ノ ハ ヒ フ ヘ ホ マ
ミ ム メ モ ヤ ユ ヨ ラ リ ル レ ロ ワ ン ゛ ゜
═  ╞  ╪  ╡  ◢  ◣  ◥  ◤  ♠  ♥  ♦  ♣  ●  ○  ╱  ╲
╳  円 年 月 日 時 分 秒
Ok
```

PRINT＃　N80　N　入出力ステートメント

機能　**シーケンシャルファイルにデータを出力します。**

書式　**PRINT＃〈ファイル番号〉,〔〈式〉…〕**

〈ファイル番号〉が1～15のときは ディスク

文例　**PRINT＃2, A, B, C**

解説

・〈ファイル番号〉は，そのシーケンシャルファイルを OPEN によってオープンしたときに指定した番号です。また，〈ファイル番号〉に－1を指定すると，カセットテープへの出力となります。なお，この場合ファイルをオープンする必要はありません。

・〈式〉はファイルに書き込まれる数値式または文字式です。PRINT ＃文を実行して，もしバッファが一杯になれば書き出し，エンドオブファイルになっていなければ次のレコードをバッファへ読み込みます。

・PRINT ＃はデータの圧縮を行わず，PRINT で画面へ出力するものと全く同じものを出力します。ですからこれらのデータがフロッピィディスクやカセットテープから正しく入力できるように，書き込むデータは適切に区切られるようにしてください。

・〈式〉のリストの中の数値式はセミコロンで区切るようにしてください。例えば，

```
PRINT ＃1, A; B; C; X; Y; Z
```

(もし区切りの記号にコンマを使うとプリント領域の間に挿入される余分な空白も媒体に書き込んでしまいます。)

　リストの中の文字列もセミコロンで区切ってください。媒体上に文字列を正しく書き込むためには，〈式〉のリストの中に独立した区切り記号を入れてください。

例をあげると，A$＝“CAMERA” そして，B$＝“93604−1” のときの次の文

PRINT ＃1, A$; B$

はディスクにCAMERA93604−1と書き込みます。これは区切り記号がありませんから2つの別の文字列として入力することはできません。この問題を解決するには，次のようにPRINT ＃中に独立に区切り記号を入れてください。

PRINT＃1, A$; “ , ” ; B$

これによってディスクに書き込まれるイメージは，

CAMERA, 93604−1

となり，2つの文字変数として読み込むことができます。

もし文字列それ自身がデータとしてコンマ，セミコロン，意味のある空白，キャリッジリターン，またはラインフィードなどを含む場合は，別の引用符，CHR$(34)によって囲んでディスクに書いてください。

例をあげると，A$＝“CAMERA, △ AUTOMATIC”, B$＝“△△△93604−1” のとき，次の文，

PRINT＃1, A$; B$

は次のようなイメージをディスクに書きます。

CAMERA, △ AUTOMATIC △△△ 93604−1

そして次の文，

INPUT＃1, A$, B$

は，“CAMERA” を A$に，そして “△ AUTOMATIC △△△ 93604−1” を B$に読み込みます。これらの文字列をフロッピィディスク上で正しく分割するには，CHR$(34) により引用符をフロッピィディスクに書き込んでください。次の文，

PRINT ＃1, CHR$ (34) ; A$; CHR$(34) ;

CHR (34) ; B$; CHR$(34)

は次の内容をディスクに書き込みます。

"CAMERA" "△ AUTOMATIC △△△ 93604−1"

すると次の文,

INPUT ＃1, A$, B$

は CAMERA, を A$ に，△ AUTOMATIC △△△ 93604−1 を B$ に読み込みます。

PRINT ＃ は，USING と共に使ってディスクファイルのフォーマットを制御することができます。

参 照　INPUT ＃, OPEN, PRINT ＃ USING

サンプルプログラム

```
100 '-- sequential data write --
110 '
120 OPEN "stest.dat" FOR OUTPUT AS #1
130 PRINT #1,DATE$;",";TIME$
140 'data write
150 INPUT "ﾋﾝﾒｲ";NA$
160 IF NA$="end" THEN 220
170 INPUT "ｶｶｸ";PL
180 INPUT "ｺｽｳ";N%
190 PRINT #1,NA$;",";PL;N%
200 GOTO 150
210 'exit 1
220 PRINT:CLOSE
230 '-- sequential data read --
240 TT=0
250 OPEN "stest.dat" FOR INPUT AS#1
260 INPUT #1,DA$,TI$
270 PRINT "ﾋｽﾞｹ : ";DA$,"ｼﾞｶﾝ : ";TI$
280 PRINT
290 'data read
300 IF EOF(1) THEN 370
310 INPUT #1,NA$,PL,N%
320 SU=PL*N%
330 PRINT NA$;TAB(10);PL;"*";N%;"=";SU
340 TT=TT+SU
350 GOTO 300
360 'exit2
370 PRINT:PRINT "TOTAL =";TT
380 END

run
ﾋﾝﾒｲ? ﾁｮｺﾚｰﾄ
ｶｶｸ? 250
ｺｽｳ? 3
ﾋﾝﾒｲ? ﾁｰｽﾞｹｰｷ
```

```
ｶｶｸ? 300
ｺｽｳ? 5
ﾋﾝﾒｲ? ﾜｲﾝ
ｶｶｸ? 3000
ｺｽｳ? 1
ﾋﾝﾒｲ? end

ﾋｽﾞｹ : 83/01/03                ｼﾞｶﾝ : 22:26:23

ﾁｮｺﾚｰﾄ        250 * 3 = 750
ﾁｰｽﾞｹｰｷ       300 * 5 = 1500
ﾜｲﾝ           3000 * 1 = 3000

TOTAL = 5250
Ok
```

PRINT% [N80] [N] 入出力ステートメント

(機能) **RS-232C 通信ポートの入出力バッファにデータを出力します。**

(書式) **PRINT%<ポート番号>, 〔<式>……〕**

(文例) **PRINT% 1, A, B**

(解説)
• <ポート番号>で指定された RS-232C 通信ポートの入出力バッファにデータを出力します。使い方はフロッピィディスクへ出力するPRINT#と同じです。

• <ポート番号>は本体内蔵の RS-232C インタフェースを使用する場合に1を指定します。

(参照) INPUT%, INIT%

(サンプルプログラム)

```
100 '-- print % test --
110 POKE &H8001,&H23:POKE &H8000,&H7F
120 OUT &HE6,4
130 OUT &HE4,&HFF
140 INIT %1,&HFA,&H37
150 LINE INPUT "message?";A$
160 IF A$="end" THEN OUT &HE6,0:END
170 PRINT %1,A$
180 GOTO 150
```

PRINT USING/LPRINT USING N80 N

入出力ステートメント

機能　**文字列，数値を指定した書式で出力します。**

書式　**{PRINT | LPRINT} USING〈書式制御文字列〉；〈式〉〔{；|，}〈式〉…〕〔{；|，}〕**

文例　**PRINT USING "＃＃＃＃,.＃"；A, B**

解説　〈書式制御文字列〉によって後に続く〈式〉の出力される領域や書式を決定します。

文字の書式制御

！……与えられた文字変数や文字定数の最初の1文字だけ出力します。

&〈n 個の△(空白)〉&……与えられた文字変数や文字定数の先頭から(n＋2)文字の文字列を出力します。与えられた文字列が(n＋2)文字よりも長い場合は，余分な文字は無視され，短い場合には，文字列は左づめに出力され，残った部分には空白が出力されます。

数値の書式制御

＃……数値を出力する桁数を指定します。指定した桁数より数値の桁数が小さいときには，右づめで出力されます。

．……小数点の位置を指定します。小数点以下の部分で冗長となる桁には0が出力されます。

＋……〈書式制御文字列〉の最初または最後に付けた場合，数値の符号がそれぞれ前または後に出力されます。2個以上の"＋"を並べた場合には，余分は後述の制御文字以外の文字と同じ扱いになります。

－……〈書式制御文字列〉の最後に付けた場合，数値が負の数の時に数値の後ろに"－"が出力されます。前に付けたり，

2個以上並べた場合には後述の制御文字以外の文字と同じ扱いになります。

＊＊……〈書式制御文字列〉の先頭につけた場合，数値領域の左側に空白部分ができたとき，そこを“＊”で埋めて出力します。この“＊＊”は2桁分の領域を確保します。

￥￥……〈書式制御文字列〉の先頭につけた場合，出力される数値の直前に“￥”を出力します。“￥”は2桁分の領域を確保しますが，このうち1桁分は“￥”の出力領域として使われます。後述の指数形式の書式指定を行った場合には，正しく出力されない事があります。

＊＊￥……〈書式制御文字列〉の先頭につけた場合，上記の2つ(＊＊と￥￥)の両方の機能となります。“＊＊￥”は3桁分の領域を確保しますが，このうち1桁分は“￥”の出力領域として使われます。

，……桁数指定の“#”の並びの中においた場合，数値の整数部が3桁毎に“,”で区切られて出力されます。但し，“,”より右側においた場合は，数値の最後に“,”が出力され，3桁毎の区切りは行われません。

^^^^……桁数指定の“#”の後に付けた場合，数値は指数形式で出力されます。

制御文字以外の文字

以上の制御文字以外の文字(英数字，カナ，グラフィック記号等)を置いた場合，数値の前や後ろにそのキャラクタが出力されます。

数値の領域をこえた場合

指定した数値領域より数値の桁数が大きい場合，数値の直前に“%”が出力されます。数値の丸めが領域より大きくなる原因となった場合も，丸めた数値の前に“%”が出力されます。

サンプルプログラム

```
100 PRINT USING "!";"NEC PC-8001MKⅡ"
110 PRINT USING "&            &";"NEC PC-8001MKⅡ"
120 PRINT USING "#####";123.456
130 PRINT USING "#####.#";123.456
140 PRINT USING "+####.#";123.456
150 PRINT USING "#####.#-";-123.456
160 PRINT USING "**###.#";123.456
170 PRINT USING "¥¥###.#";123.456
180 PRINT USING "**¥##.#";123.456
190 PRINT USING "#####,.#";1234.56
200 PRINT USING "#####";123456!
210 END

run
N
NEC PC-8001MKⅡ
  123
  123.5
 +123.5
  123.5-
**123.5
 ¥123.5
*¥123.5
 1,234.6
%123456
Ok
```

PRINT♯ USING [N80] [N] 入出力ステートメント

(機能) 文字列，数値を指定した書式でファイルに出力します。

(書式) PRINT♯〈ファイル番号〉, USING〈書式制御文字列〉

; 〈式〉[; 〈式〉…][;]
　　　 ,　　　　 ,

〈ファイル番号〉が1～15のときは [ディスク]

(文例) PRINT♯2, USING "♯♯♯♯"; A

(解説) • 〈ファイル番号〉で指定されたファイルに対して，文字列や数値を書式指定して出力します。

• PRINT ♯ USING は，その対象がファイルであることを除けば PRINT USING と全く同じです。

(参照) PRINT USING, PRINT ♯, OPEN

(サンプルプログラム)

```
100 OPEN "stest.dat" FOR OUTPUT AS 1
110 FOR I=0 TO 10
120   PRINT #1,USING "###";I^2
130 NEXT I
140 CLOSE
150 END
```

PSET [N80] [N] グラフィックステートメント

機能 **低解像度グラフィック座標(Lx,Ly)上のピクセルをセットします。**

書式 **PSET(Lx, Ly**[, 〈ファンクションコード〉])

文例 **PSET(100, 50, 7)**

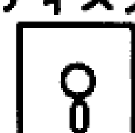
解説
- 低解像度グラフィック座標(Lx, Ly)にピクセルをセットします。
- オプションの〈ファンクションコード〉はCOLORに用いる場合と同じで，カラーモードの時は色を指定することができ，白黒モードの時はリバースなどの機能を指定することができます。

注意 PSETは，COLORで指定される〈グラフィックスイッチ〉がONになっている状態で使用してください。

参照 PRESET, COLOR, CONSOLE

サンプルプログラム

```
100 WIDTH 80,25
110 CONSOLE ,,,1
120 CMD CLS 1
130 FOR I=0 TO 90
140   PSET(I,I,C)
150   C=I MOD 7
160 NEXT I
```

PUT ディスク    入出力ステートメント

機能 バッファ中のデータをランダムアクセスファイルに書き出します。

書式 PUT〔#〕<ファイル番号>〔,<数式>〕

文例 PUT#3, 5

解説 • PUT は<ファイル番号>で指定されたバッファの内容をランダムファイルへの書き込みとして働きます。指定されたファイルは，ランダムファイルのモードでオープンされていなければなりません。

• <数式>はファイルのレコード番号として解釈され，バッファ中のデータ256バイトが指定されたレコードに書き込まれます。レコード番号が省略された場合，直前に行われたGET, PUT で参照されたレコードの次のレコードに書き込まれます。なお，書き出すデータは予め FIELD, LSET/RSET により準備しておかねばなりません。

参照 OPEN, FIELD, GET, LSET/RSET, PC-8001MKⅡ ユーザーズマニュアル第8章

サンプルプログラム

```
100 '-- random data write --
110 OPEN "1:data1" AS #1
120 FIELD #1,30 AS A$,20 AS B$,50 AS C$
130 'entry
140 LINE INPUT "NAME?";NA$
150 IF NA$="end" THEN 240
160 LINE INPUT "TEL?";TL$
170 LINE INPUT "ADDRESS?";AR$
180 LSET A$=NA$
190 LSET B$=TL$
200 LSET C$=AR$
210 PUT #1
220 GOTO 140
230 'exit
240 CLOSE
run
NAME?ｺﾏﾂ ｼｹﾞｷ
TEL?01-234-5678
ADDRESS?ﾄｳｷｮｳﾄ ﾐﾅﾄｸ
NAME?ﾀｹﾀﾞ ﾋﾛｼ
TEL?098-765-4321
```

```
ADDRESS?ｶﾅｶﾞﾜｹﾝ ﾖｺﾊﾏｼ
NAME?ﾀｶｷﾞ
TEL?124-816-3264
ADDRESS?ｵｵｻｶﾌ ｵｵｻｶｼ
NAME?end
Ok
```

PUT@　N80 N　グラフィックステートメント

機能　GET @で配列へ読み込んだキャラクタや低解像度グラフィックパターンを画面の任意の位置へ表示します。

書式　1） PUT@(| X1, Y1 / Lx1, Ly1 |)−(| X2, Y2 / Lx2, Ly2 |), <配列名>, <機能>

2） PUT@A(X1, Y1)−(X2, Y2), <配列名>

文例　PUT@(20, 20)−(30, 24), G%, PSET

解説　1) GET @でキャラクタとして配列へセーブしたものを表示させる場合水平位置X1, X2は0〜(1行の桁数)−1まで，垂直位置 Y1,Y2 は 0 〜(画面の行数)−1までの値をとります。そして<機能>には色やブリンクなどを指定するための<ファンクションコード>を置くことができます。

GET@で低解像度グラフィックパターンとして配列へセーブしたものを表示させる場合の Lx_1, Lx_2 は0から(1行の桁数)＊2−1までの値，Ly_1, Ly_2 は0〜(画面の行数)＊4−1までの値をとります。そして<機能>には PSET, PRESET, OR, AND, NOT, XOR のいずれかを指定しなければなりません。OR, AND, XOR は配列のデータと画面上に表示されているものとの間で論理演算を行います。論理値は表示されているピクセルが1となり配列中のデータもピクセルがあるところが1となります。NOT は配列内のデータを反転して表示します。

GET@Aでセーブした配列をAの付かないこの書式で表示させることはできません。

2) GET @文の書式3) の方法でセーブしたものを表示させる。

水平位置 X1, X2 は0から(1行の桁数)−1までの値，垂直位置 Y1, Y2 は0から(画面の行数)−1までの値をとります。色などもセーブした時点のものを自動的に与えます。

GET @文の書式1) でセーブしたものをこの書式で表示することはできません。

参照　GET @

サンプルプログラム

```
100 DIM G%(100)
110 LINE(0,0)-(10,10),PSET,BF
120 GET@(0,0)-(10,10),G%,G
130 PUT@(100,50)-(110,60),G%,PSET
```

READ　N80 N　一般ステートメント

機能　**DATA 文より値を読み，変数に割り当てます。**

書式　**READ〈変数名リスト〉**

文例　**READ K, M, S$**

解説

• READ はいつでも DATA と組み合わせて使わなければなりません。READ は DATA のデータを 1 対 1に対応させて変数に割りあてていきます。READ の変数は数値変数でも文字変数でもかまいません。しかし読み出された値と変数の型は一致していなければなりません。もし，型が一致しない場合には，“Syntax error” が起ります。

• 1 つの READ が 1 つまたはそれ以上の DATA を参照したり(複数の場合は順番に参照されます)，またいくつかの READ が 1 つの DATA を参照することができます。もし〈変数名リスト〉の変数の数が DATA のデータの個数を越えてしまった場合は，“Out of DATA” のエラーメッセージが表示されます。指定された変数が DATA のデータの個数よりも少ない場合には，その次の READ が読まれなかったデータから読み始めます。もしそれ以上 READ がない場合には，余分のデータは無視されます。

• 複数の DATA から READ によって読む DATA を指定するには RESTORE を使います。

参照　RESTORE, DATA

サンプルプログラム

```
100 FOR I=0 TO 10
110   READ A
120   BEEP 1:BEEP 0
130   FOR J=1 TO 500:NEXT J
140   PRINT A;
150 NEXT I
160 READ A$:PRINT A$
```

```
170 FOR I=1 TO 50
180  BEEP 1
190  FOR J=I TO 50
200    BEEP 0
210 NEXT J,I
220 END
230 DATA 10,9,8,7,6,5,4,3,2,1,0,"    FIRE!"

run
 10  9  8  7  6  5  4  3  2  1  0     FIRE!
Ok
```

REM [N80] [N] 一般ステートメント

機能 プログラム中の注釈です。

書式 REM〔<注釈文>〕

文例 REM＊＊TEST PROGRAM＊＊

'＊＊TEST PROGRAM＊＊

解説

• REM は，ユーザがプログラムを見やすくしたりコメントを書いたりするために使われ，プログラムの実行には全く影響をおよぼしません。

• REM は GOTO, GOSUB の飛先の目印として使うことができます。

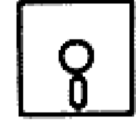
• キーワード "REM" の代わりにアポストロフィー(')を使うことができます。

• REM では，コロン(:)も注釈の一部となりますので，コロンで区切って他の文を続けることはできません。

• REM は BASIC によって実行される文の後にコロンで区切って付け加えることができます。

サンプルプログラム

```
100 '      REM ｽﾃｰﾄﾒﾝﾄ ﾆ ﾖｯﾃ
110 REM   ﾌﾟﾛｸﾞﾗﾑ ﾉ ﾅｶﾆ
120 REM   ｺﾒﾝﾄ ｶﾞ ｶｹﾏｽ｡
130 REM
140 '      REM ﾊ ｺﾉ ﾚｲ ﾉ ﾖｳﾆ
150 '     "'" ﾃﾞ ﾀﾞｲﾖｳ ﾃﾞｷﾏｽ｡
```

REMOVE   入出力コマンド

ディスク

機能　フロッピィディスクに配置表(FAT)を書き込んでフロッピィディスクの入出力を終了します。

書式　REMOVE〔<ドライブ番号>〔,<ドライブ番号>……〕〕

文例　REMOVE 1

解説
• REMOVEはMOUNTの逆の動作をします。フロッピィディスクをドライブから取り出す前に必ず実行しなければなりません。

• REMOVEを実行すると現在メモリ上にある配置表(FAT)を，コピーと合わせて3組，フロッピィディスクのディレクトリトラックに書き込み，MOUNTと同じように誤りを検出します。

注意
• フロッピィディスクの入出力を終了し，ドライブからフロッピィディスクを取り出すときは，必ずREMOVEを実行してください。

もし，REMOVEの実行をしなかった場合，取り出したフロッピィディスクの配置表(FAT)は更新と誤り検出が行われず，またそのドライブに入れられた次のディスケットには誤った配置表が書き込まれてしまい両方のフロッピィディスクの配置表(FAT)が破壊されてしまいます。

• N80 DISK-BASICではオートマチックMOUNT，REMOVEを採用しているため，これらの操作はすべてBASICが管理します。ただしN80 DISK-BASICでREMOVEを実行してもエラーは起りません。

参照　MOUNT

サンプルプログラム

```
remove 1
Ok
remove
Ok
```

RENUM N80 N 一般コマンド

機能 **プログラムの行番号を整理します。**

書式 **RENUM〔<新行番号>〕〔,<旧行番号>〕〔,<増分>〕**

文例 **RENUM 1000, 100**

解説

• 〈新行番号〉は，新しくつける行番号の最初の行番号で，省略したときは10です。

• 〈旧行番号〉は，行番号のつけ替えをはじめる現在のプログラム行番号です。省略したときはそのプログラムの最初の行番号です。

• 〈増分〉は新しく付ける各行番号のあいだの増分で，省略したときは10です。

RENUM は，GOTO, GOSUB, ON … GOTO, ON … GOSUB 及び ERL などで参照している行番号も新しい行番号に対応して変更します。これらの文が参照している行番号の行が RENUM を実行する前のプログラム中に存在しない場合には，“Undefined Line xxxxx in yyyyy” のエラーメッセージが表示されます。この場合，存在しなかった行番号 (xxxxx) は RENUM によって変更されませんが，その文の新しい行番号 (yyyyy) は変更されます。

注意

RENUM はプログラム行の順序を入れ変えるのに使用することはできません。(たとえば，10, 20, 30の 3 つの行がある場合，RENUM 15, 30で10, 15(もとの30行)，20とすることはできません)。また65529以上の行番号を発生することもできません。このような場合 “Illegal function call” エラーが起こります。

サンプルプログラム

```
list
100 ' RENUM ｺﾏﾝﾄﾞ ﾊ ｼﾃｲｼﾀ ｷﾞｮｳ ｶﾗ
110 ' ｷﾞｮｳ ﾊﾞﾝｺﾞｳ ｦ ﾂｹｶｴ ﾏｽ｡
120 '
130 PRINT " RENUM ｺﾏﾝﾄﾞ ﾊ ｼﾃｲｼﾀ ｷﾞｮｳ ｶﾗ"
140 PRINT " ｷﾞｮｳ ﾊﾞﾝｺﾞｳ ｦ ﾂｹｶｴ ﾏｽ｡"
150 END
Ok
renum 200,130,20
Ok
list
100 ' RENUM ｺﾏﾝﾄﾞ ﾊ ｼﾃｲｼﾀ ｷﾞｮｳ ｶﾗ
110 ' ｷﾞｮｳ ﾊﾞﾝｺﾞｳ ｦ ﾂｹｶｴ ﾏｽ｡
120 '
200 PRINT " RENUM ｺﾏﾝﾄﾞ ﾊ ｼﾃｲｼﾀ ｷﾞｮｳ ｶﾗ"
220 PRINT " ｷﾞｮｳ ﾊﾞﾝｺﾞｳ ｦ ﾂｹｶｴ ﾏｽ｡"
240 END
Ok
```

RESTORE N80 N 一般ステートメント

(機能) READで読むDATAを指定します。

(書式) RESTORE〔〈行番号〉〕

(文例) RESTORE 800

(解説) • 〈行番号〉を指定すると，指定された行以降のDATAの内容から読み始めます。〈行番号〉を省略したときは，プログラム中の最初のDATAの内容から読み始めます。

(サンプルプログラム)

```
100  READ A,B,C
110 RESTORE
120  READ D,E,F
130 RESTORE 220
140  READ A$,B$,C$
150  PRINT A,B,C
160  PRINT D,E,F
170  PRINT A$,B$,C$
180 END
190  DATA 1,2,3
200  DATA 4,5,6
210 'string data
220  DATA aa,bb,cc

run
 1             2             3
 1             2             3
aa            bb            cc
Ok
```

RESUME　N80 N　特殊ステートメント

(機　能)　**エラー回復処理終了後，プログラムの実行を再開します。**

(書　式)　**RESUME** | 〔0〕 / NEXT / 〈行番号〉 |

(文　例)　**RESUME 0**

RESUME NEXT

RESUME 1000

(解　説)　• RESUME は，ON ERROR GOTO によってエラー処理ルーチンにジャンプしたあと，プログラムの実行を再開する場合に用います。

• プログラムの実行を再開する場所に応じて次のように3つに書式を選ぶことができます。

RESUME 〔0〕 ………… エラーの原因となった文からプログラムの実行が再開されます。また，このときエラーとなった原因を取り除かないと再びエラー処理ルーチンへ戻ります。

RESUME NEXT ………… エラーの原因となった文のすぐ次の文から実行が再開されます。

RESUME 〈行番号〉 ……… 〈行番号〉で指定した行から実行が再開されます。

(参　照)　ON ERROR GOTO

(サンプルプログラム)

```
100 ON ERROR GOTO 170
110 'entry
120 INPUT "X,Y";X,Y
130 A=X/Y
140 PRINT "X/Y =";A
150 GOTO 120
160 'errsub
170 PRINT "** ｹｲｻﾝ ﾃﾞｷﾏｾﾝ **"
180 RESUME 120
```

```
run
X,Y? 100,10
X/Y = 10
X,Y? 35,0
** ｹｲｻﾝ ﾃﾞｷﾏｾﾝ **
X,Y? 35,5
X/Y = 7
X,Y?
Break in 120
Ok
```

RIGHT$ N80 N 文字関数

機能 **文字列の右側から指定された長さの文字列を与えます。**

書式 **RIGHT$**(〈**文字列**〉, 〈**数式**〉)

文例 **B$＝RIGHT$(A$, 3)**

解説
• RIGHT＄は〈文字列〉の右側から〈数式〉で指定された長さの文字列を取り出す関数です。
• 〈数式〉は０～255の範囲になければなりません。
• 〈数式〉が，〈文字列〉の総文字数以上の場合は，〈文字列〉全体を，また，〈数式〉が０ならヌルストリングを与えます。

参照 LEFT$, (2)MID$

サンプルプログラム

```
100 ' ﾓｼﾞﾚﾂ(10) ﾉ ｶｲﾃﾝ
110 INPUT A$
120 FOR I=1 TO 11
130   PRINT A$
140   B$=RIGHT$(A$,1)
150   A$=B$+MID$(A$,1,LEN(A$)-1)
160 NEXT I
170 END

run
? 9876543210
9876543210
0987654321
1098765432
2109876543
3210987654
4321098765
5432109876
6543210987
7654321098
8765432109
9876543210
Ok
```

RND [N80] [N] 数値関数

(機 能) 乱数を与えます。

(書 式) RND(〈数式〉)

(文 例) A=RND(1)*5

(解 説) • 0以上1未満の乱数を与えます。

• 発生される乱数は〈数式〉の値によって次のように異なります。

〈数式〉が負の場合——新しい乱数系列を設定します。

〈数式〉が0の場合——1つ前に発生した乱数の値をとります。

〈数式〉が正の場合——次の乱数を発生します。

(サンプルプログラム)

```
100 '-- ﾗﾝｽｳ ｹﾝﾃｲ --
110 DIM SU(6)
120 FOR I=1 TO 100
130   DA=INT(RND(1)*6+1)
140   SU(DA)=SU(DA)+1
150 NEXT I
160 FOR I=1 TO 6
170   PRINT I"-";SU(I);"%"
180 NEXT I

run
 1 - 22 %
 2 - 15 %
 3 - 13 %
 4 - 12 %
 5 - 17 %
 6 - 21 %
Ok
```

RUN　N80 N　一般コマンド

機能　1)メモリ上にあるプログラムの実行を開始します。

2)フロッピィディスクからファイルをメモリにロードし，そのプログラムを実行します。

書式　1)　RUN〔<行番号>〕

2)　RUN<ファイル名>〔,R〕 ディスク

文例　1) RUN 100

2)　RUN "2 : sample"

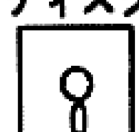
解説　1) <行番号>を指定すると，その行から実行がはじまります。指定のない場合には，最も若い行番号の行から実行がはじまります。プログラムの実行が終るとBASICはコマンドレベルに戻ります。

2) (N80) DISK-BASICでは1)の機能に加えて，メモリ上にプログラムがない場合も<ファイル名>で指定されるプログラムをロードした後，実行することができます。

• RUNを実行すると，すべての開いているファイルは閉じ，目的のプログラムをロードする前にメモリの内容をクリアします。しかし，Rオプションを付けた場合には，すべてのデータファイルは開いたままになります。

サンプルプログラム

```
run
run 1000
run "sample"
run "2:test",r
```

SAVE    入出力コマンド

機能　メモリ上の BASIC プログラムをフロッピィディスクにセーブします。

書式　SAVE<ファイル名>, 〔A〕

文例　SAVE "sample", A

解説

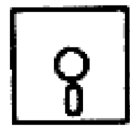
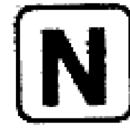
• 〈ファイル名〉で指定されるファイルにプログラムを書き込みます。指定したファイル名と同じ名前のファイルが存在した場合には，古い内容は失なわれて，新しいものに更新されます。

• A オプションが指定された場合には，プログラムはアスキー型式でセーブされます。オプションの指定がない場合には，バイナリ型式に圧縮されてプログラムのセーブが行われます。アスキー型式のセーブは，バイナリ型式よりも，多くのファイルスペースを必要としますが，セーブされたプログラムファイルを操作する場合には，アスキー型式でセーブされていなければなりません。例えば MERGE コマンドは，アスキー型式のファイルを必要とします。また，アスキー型式でセーブされたファイルは，データファイルとして読み出すことができます。

参照　LOAD, MERGE

サンプルプログラム

```
save "2:Space.Ody"
Ok
```

SET    入出力ステートメント

機能　**ファイルの属性をセットあるいはリセットします。**

書式　**1) SET〈ドライブ番号〉, 〈"属性文字"〉**

2) SET〈ファイル名〉, 〈"属性文字"〉

3) SET＃〈ファイル番号〉, 〈"属性文字"〉

文例　**1) SET 1, "P"**

2) SET "2 : NOTWRITE", "P"

3) SET＃1, "R"

解説　• 指定したドライブに入っているフロッピィディスク，ファイル番号，ファイルに，〈"属性文字"〉によって指定された属性を付けます。属性文字は，**大文字の P, R** で指定し，それぞれ，ライトプロテクト(書き込み禁止)，リードアフターライト(書き込み確認)の属性を付けます。これ以外の文字が属性文字として指定された場合には，現在設定されている属性がすべて解除されます。

1)〈ドライブ番号〉が指定された場合には，指定されたドライブの中に入っているフロッピィディスクに対して属性がつけられます。

2)〈ファイル名〉が指定された場合には，指定されたファイルのみに属性がつけられ，同一のフロッピィディスク中の他のファイルにはその影響は及びません。

3)〈ファイル番号〉が指定された場合には，以後そのファイルへの出力に対して指定された属性が作用します。

• P 属性を付けると，PRINT＃, PUT, 等の書き込み動作が禁止されるとともに，ファイルの KILL, RENAME もできなくなります。

• R 属性を付けると，書き込みを行った直後に読み出しを

行い正しく書き込みが行れたかの確認がされるようになります。

参照　ATTR$

サンプルプログラム

```
save "1:sample"
Ok
set 1,"P"
Ok
kill "1:sample"
File write protected
Ok
```

SGN　[N80] [N]　数値関数

(機　能) 数値の符号を与えます。

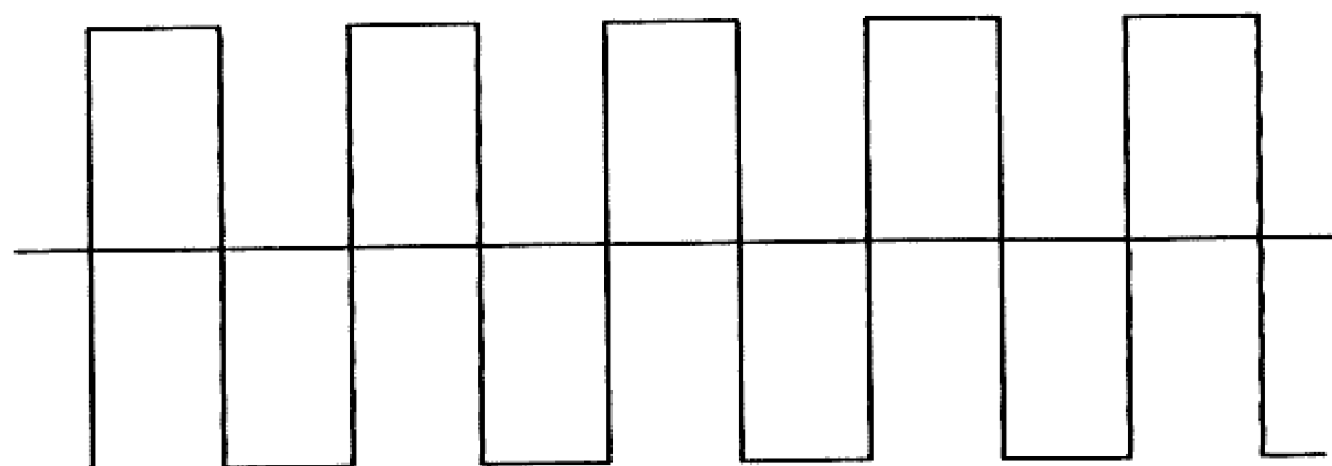
(書　式) SGN(<数式>)

(文　例) PRINT SGN(A)

(解　説) • <数式>が正の場合1を，0の場合は0を，負の場合は−1を与えます。

(サンプルプログラム)

```
100 CMD SCREEN 2,,5:CMD CLS 3
110 CMD LINE(0,100)-(319,100),2
120 X=5
130 FOR I=0 TO 31.42 STEP .05
140   S=SGN(INT(COS(I)*20))
150   ON S+2 GOSUB 200,220,240
160   X=X+1
170 NEXT I
180 END
190 'plus
200 CMD PSET (X/2,20),3:RETURN
210 'zero
220 CMD LINE(X/2,20)-(X/2,179),3:RETURN
230 'minus
240 CMD PSET (X/2,179),3:RETURN
```



SIN　N80 N　数値関数

機能　正弦(サイン)の値を与えます。

書式　SIN(<数式>)

文例　SIN(3.14159/22)

解説

- 〈数式〉の単位をラジアン(π/180　×角度)としたときの正弦(サイン)を与えます。
- SIN 関数の演算は単精度で行われます。

サンプルプログラム

```
100 '-- csec ｦ ﾓﾄﾒﾙ --
110 PI=3.14159
120 F$="csc(###°)=###.#####
130 INPUT "ｶｸﾄﾞ :";D
140 R=D MOD 180
150 IF R=0 THEN PRINT "ﾃｲｷﾞ ｻﾚﾏｾﾝ":END
160 C=1/SIN(R/180*PI)
170 PRINT USING F$;D,C
180 END
```

```
run
ｶｸﾄﾞ :? 15
csc( 15°)=  3.86371
Ok
run
ｶｸﾄﾞ :? 60
csc( 60°)=  1.15470
Ok
```

SPACE$ [N80] [N] 文字関数

機能 指定した長さの空白文字列を与えます。

書式 SPACE$(〈数式〉)

文例 PRINT SPACE$(10) ; "ABC"

解説 • 〈数式〉の数だけの空白をもった文字列を与えます。〈数式〉の範囲は 0 ～255までの範囲になければなりません。

参照 SPC

サンプルプログラム

```
100 '-- squre curve --
110 FOR I=-6 TO 6
120   A$=CHR$(&H88)+SPACE$(ABS(I*I))+"*"
130   PRINT A$
140 NEXT I
150 END
```

```
run
|                                    *
|                         *
|                *
|         *
|    *
| *
|*
| *
|    *
|         *
|                *
|                         *
|                                    *
Ok
```

SPC　N80 N　文字関数

機　能　指定した数だけ空白を出力します。

書　式　**SPC**(〈数式〉)

文　例　**PRINT SPC(10) ; A$**

解　説

- SPC 関数は PRINT や LPRINT など出力文中でのみ使用することができます。
- 〈数式〉の値は，0 ～255の範囲です。

参　照　SPACE$, TAB

サンプルプログラム

```
100 FOR I=0 TO 10
110   PRINT "N80-BASIC";SPC(I);"N-BASIC"
120 NEXT I
130 END

run
N80-BASICN-BASIC
N80-BASIC N-BASIC
N80-BASIC  N-BASIC
N80-BASIC   N-BASIC
N80-BASIC    N-BASIC
N80-BASIC     N-BASIC
N80-BASIC      N-BASIC
N80-BASIC       N-BASIC
N80-BASIC        N-BASIC
N80-BASIC         N-BASIC
N80-BASIC          N-BASIC
Ok
```

SQR　N80　N　数値関数

(機能) 平方根(スクエアルート)を与えます。

(書式) SQR(〈数式〉)

(文例) A=SQR(2)

(解説)
- 〈数式〉で指定された値の平方根を与えます。
- SQR 関数の演算は単精度で行われます。

(サンプルプログラム)

```
100 '-- square root table --
110 PRINT " << SQUARE ROOT TABLE >>"
120 FOR N=0 TO 10
130   PRINT "N =";N,"ROOT(N) =";
140   PRINT USING "##.#####";SQR(N)
150 NEXT N

run
 << SQUARE ROOT TABLE >>
N = 0          ROOT(N) =  0.00000
N = 1          ROOT(N) =  1.00000
N = 2          ROOT(N) =  1.41421
N = 3          ROOT(N) =  1.73205
N = 4          ROOT(N) =  2.00000
N = 5          ROOT(N) =  2.23607
N = 6          ROOT(N) =  2.44949
N = 7          ROOT(N) =  2.64575
N = 8          ROOT(N) =  2.82843
N = 9          ROOT(N) =  3.00000
N = 10         ROOT(N) =  3.16228
Ok
```

STATUSPOINT N80 グラフィック関数

機能

1) Last Referenced Point (LP) の値を与えます。

2)ビュー座標上の指定された座標に表示されているピクセルのカラーナンバを与えます。

書式

1) STATUSPOINT(〈機能〉)

2) STATUSPOINT(Hx, Hy)

文例

1) HX=STATUSPOINT(0)

2) CL=STATUSPOINT(100, 100)

解説

1) 最後に高解像度グラフィック操作の行われた座標 (LP) を〈機能〉の指定により，ビュー座標で与えます。
〈機能〉は0と1の整数値をとり STATUSPOINT の返す値は次のようになります。

0：最後に高解像度グラフィック操作の行われた点の X 座標(ビュー座標系)を返します。

1：同じく Y 座標を返します。

2) (Hx, Hy) により指定されたビュー座標上に表示されているピクセルの色をカラーナンバで返します。また指定された座標 (Hx, Hy) がビューポート外にある場合は－1を返します。

サンプルプログラム

```
10 '  POINT function
20 CMD SCREEN 2
30 CMD CLS 2
40 CMD POINT(10,20)
50 IF STATUSPOINT(0)=10 THEN PRINT "Good POINT function"
60 IF STATUSPOINT(1)=20 THEN PRINT "Good POINT function"
70 CMD PSET(100,110),3
80 IF STATUSPOINT(100,110)=3 THEN PRINT "Good POINT function"
90 END
```

STATUSVIEW　N80　グラフィック関数

機能　**現在のビューポートの設定位置を与えます。**

書式　**STATUSVIEW**(〈**機能**〉)

文例　**STATUSVIEW(1)**

解説　• CMD VIEWにより設定されている現在のビューポートの位置(Hx_1,Hy_1)−(Hx_2,Hy_2)を返す関数です。〈機能〉は0〜3までの整数値をとりSTATUSVIEWの返す値は次のようになります。

0：ビューポートの左右の頂点のX座標(Hx_1)

1：ビューポートの左右の頂点のY座標(Hy_1)

2：ビューポートの右下の頂点のX座標(Hx_2)

3：ビューポートの右下の頂点のY座標(Hy_2)

• STATUS VIEW関数は，画面上のビューポートの位置を返す関数ですから，絶対座標系の値で返されます。

参照　CMD VIEW, 1章ビューポート

サンプルプログラム

```
10 '  VIEW function
20 CMD VIEW(10,20)-(30,40)
30 IF STATUSVIEW(0)=10 THEN PRINT "Good VIEW function"
40 IF STATUSVIEW(1)=20 THEN PRINT "Good VIEW function"
50 IF STATUSVIEW(2)=30 THEN PRINT "Good VIEW function"
60 IF STATUSVIEW(3)=40 THEN PRINT "Good VIEW function"
70 END
```

STR$ N80 N 文字関数

機能 数値を表わす文字列を与えます。

書式 STR$(〈数式〉)

文例 A$=STR$(123)

解説 • 〈数式〉によって指定された値を文字に変換します。但し，数値および変数以外を指定してはいけません。〈数式〉の値は整数型，実数型，（倍精度，単精度）およびE, Dを伴った指数表現など，すべての数値型に有効です。

• STR$はVALと逆の変換を行います。

注意 STRING$とSTR$を混同しないようにしてください。

参照 VAL

サンプルプログラム

```
100 ' 00:00:00 カラ シテイシタ ビョウスウ ダケ
110 ' ケイカシタ ジコク ヲ モジレツ デ モトメル
120 DEF FNST$(A$)=RIGHT$("0"+RIGHT$(A$,LEN(A$)-1),2)
130 INPUT "秒 スウ :";SS
140 H=SS¥3600:M=(SS-H*3600)¥60
150 S=SS MOD 60
160 H$=STR$(H):M$=STR$(M):S$=STR$(S)
170 H$=FNST$(H$):M$=FNST$(M$):S$=FNST$(S$)
180 PRINT "00:0:00 カラ ";SS;"秒 ケイカシタ";
190 PRINT " ジコク ハ ";H$;":";M$;":";S$; " デス"
200 END

run
秒 スウ :? 10000
00:0:00 カラ  10000 秒 ケイカシタ ジコク ハ 02:46:40 デス
Ok
```

STRING$ N80 N 文字関数

機能 指定した文字を指定した数だけ与えます。

書式 STRING$(〈式〉, |〈文字列〉 / 〈数式〉|)

文例 A$=STRING$(10, "+")

解説 • 与える文字は,〈文字列〉または〈数式〉によって指定します。〈文字列〉の場合最初の一文字が有効です。〈数式〉の場合は,値が, 0～255の範囲に限られ,この値をキャラクタコードとみなし,対応する文字を採用します。

サンプルプログラム

```
100 INPUT A$
110 L=LEN(A$)
120 PRINT STRING$(L+4,"+")
130 PRINT "+ ";SPC(LEN(A$)+1);"+"
140 PRINT "+ ";A$;" +"
150 PRINT "+ ";SPC(LEN(A$)+1);"+"
160 PRINT STRING$(L+4,"+")
170 END

run
? STRING statement sample program
++++++++++++++++++++++++++++++++++++
+                                  +
+ STRING statement sample program +
+                                  +
++++++++++++++++++++++++++++++++++++
Ok
```

STOP　N80 N　一般ステートメント

(機　能) **プログラムの実行を停止してコマンドレベルに戻ります。**

(書　式) **STOP**

(文　例) **STOP**

(解　説) • STOP はプログラムの実行を停止するもので，プログラム中のどこに置いてもかまいません。

• STOP を実行すると，次のメッセージがプリントされます。

Break in XXXXX
(XXXXX はストップした行番号)

• END と異なり，STOP はファイルをクローズしません。

• STOP が実行されると，BASIC は常にコマンドレベルに戻ります。また，CONT によりプログラムの実行が再開されます。

• STOP は，プログラムの実行中 STOP キーを押すことと全く同じ動作をします。ただし STOP キーを押して実行を停止した場合は CONT によって実行の再開ができない場合があります(“Can't continue”エラー)。

(参　照) CONT

(サンプルプログラム)

```
100 FOR I=10 TO 19
110   PRINT I,SQR(I)
120   STOP
130 NEXT I
140 END

run
 10            3.16228
Break in 120
Ok
cont
 11            3.31663
Break in 120
Ok
cont
 12            3.4641
Break in 120
Ok
```

SWAP　N88 N　一般ステートメント

機能　2つの変数の値を交換します。

書式　**SWAP〈変数名1〉，〈変数名2〉**

文例　**SWAP A$, B$**

解説　• どの型の変数(整数，単精度，倍精度，文字，単純変数と配列変数にかかわらず)でも，SWAP によりその値を交換することができます。しかし，その2つの変数の型が一致していないと "Type mismatch" エラーが起ります。

• 〈変数名2〉が単純変数(配列変数以外の変数)のとき，その変数には値が定義されていなければなりません。

サンプルプログラム

```
100 '-- 2ツ ノ ハイレツ ノ ナイヨウ ヲ イレカエル --
110 DIM A(10),B(10)
120 FOR I=1 TO 10
130   A(I)=I*2:B(I)=I*3:GOSUB 200
140 NEXT I:LOCATE 0,Y+3
150 FOR I=1 TO 10
160   SWAP A(I),B(I):GOSUB 200
170 NEXT I:LOCATE 0,Y+2
180 END
190 'print tab
200 X=POS(0):Y=CSRLIN
210 LOCATE X,Y:PRINT USING "###";A(I);
220 LOCATE X,Y+1:PRINT USING "###";B(I);
230 LOCATE X+5,Y
240 RETURN
run
  2    4    6    8  10  12  14  16  18  20
  3    6    9  12  15  18  21  24  27  30

  3    6    9  12  15  18  21  24  27  30
  2    4    6    8  10  12  14  16  18  20
Ok
```

TAB　N80 N　文字関数

機能　指定された桁位置まで空白を出力します。

書式　TAB (〈数式〉)

文例　PRINT TAB(10) ; "N80-BASIC"

解説

- TAB 関数は PRINT や LPRINT など出力文中のみで使用されます。
- 〈数式〉の値は 0 ～255までの範囲です。
- 〈数式〉が 1 行あたりの桁数よりも大きくなったときは自動的に改行されます。

参照　SPC

サンプルプログラム

```
100 FOR DG=0 TO 180 STEP 20
110   RD=DG/180#*3.14159265358979#
120   PRINT "DEG =";DG;
130   PRINT TAB(15)"SIN(DG) =";SIN(RD);
140   PRINT TAB(40)"COS(DG) =";COS(RD)
150 NEXT DG

run
DEG = 0        SIN(DG) = 0              COS(DG) = 1
DEG = 20       SIN(DG) = .34202         COS(DG) = .939693
DEG = 40       SIN(DG) = .642788        COS(DG) = .766044
DEG = 60       SIN(DG) = .866025        COS(DG) = .5
DEG = 80       SIN(DG) = .984808        COS(DG) = .173648
DEG = 100      SIN(DG) = .984808        COS(DG) =-.173648
DEG = 120      SIN(DG) = .866025        COS(DG) =-.5
DEG = 140      SIN(DG) = .642788        COS(DG) =-.766044
DEG = 160      SIN(DG) = .34202         COS(DG) =-.939693
DEG = 180      SIN(DG) = 0              COS(DG) =-1
Ok
```

TAN　N80 N　数値関数

機能　正接(タンジェント)を与えます。

書式　TAN(〈数式〉)

文例　B=TAN(C/A)

解説
- 〈数式〉の単位をラジアン(π/180×角度)としたときの正接(タンジェント)を与えます。
- TAN 関数の演算は単精度で行われます。

注意　TAN関数は

TAN (〈数式〉)=SIN(〈数式〉)/COS(〈数式〉)

という式で算出しているためCOS関数の値が0となるとき，(つまり〈数式〉の値が，　π/2,　3π/2,　5π/2 …　のとき) "Division by zero"エラーが起こります。

サンプルプログラム

```
100 '-- cot ヲ モトメル --
110 PI=3.14159
120 F$="cot(###° )=###.#####
130 INPUT "カクド":";D
140 R=D MOD 180
150 IF R=0 THEN PRINT "テイギ サレマセン":END
160 C=1/TAN(R/180*PI)
170 PRINT USING F$;D,C
180 END

run
カクド:? 45
cot( 45° )=  1.00000
Ok
run
カクド:? 30
cot( 30° )=  1.73205
Ok
run
カクド:? 23
cot( 23° )=  2.35586
Ok
```

TERM [N80] [N] 特殊コマンド

(機　能) MKⅡ をターミナルモードにします。

(書　式) TERM〈データのビット長〉，〈パリティ〉，〈ボーレート〉，〈フィードスイッチ〉

(文　例) TERM J, 2, 1, 1

(解　説) • PC-8001MKⅡ を BASIC モードからターミナルモードに変化させます。

• ターミナルモードでは，RS-232C インターフェースを介して，他のコンピュータや周辺機器との通信が可能となります。

• 〈データのビット長〉，〈パリティ〉，〈ボーレート〉，〈フィードスイッチ〉は次のようになっています。

〈データのビット長〉	a ：8 ビットモード j ：7 ビットモード
〈パリティ〉	0 ：パリティなし 1 ：奇数パリティ 2 ：偶数パリティ
〈ボーレートファクタ〉	0 ：×64 1 ：×16
〈フィードスイッチ〉	0 ：オート L/F しない 1 ：オート L/F する

• 〈データのビット長〉は 8 ビットモードの場合最上位ビットでカナと英文字の切替えを行い，7 ビットモードでは SI(shift in), SO(shift out) コードで英文字とカナ文字の切り替えが行われます。

• 〈パリティ〉はパリティの形式，有無を指定します。

• 〈ボーレートファクタ〉と，本体後部のジャンパスイッチによって実際のボーレートを決定します。ボーレートファクタが×16のとき後部ジャンパスイッチの値がそのまま採用され

ます。×64のときはジャンパスイッチの値の4分の1が実際のボーレートとなります。

•〈フィードスイッチ〉の指定を1にするとCRコードを受信したときにCR(キャリジリターン)とLF(ラインフィード)を行い改行します。0のときはCR, LFコードがそれぞれ独立して働きます。

• ターミナルモードではファンクションキーの6～10が次のような意味を持っています。

f・6：コントロールコードの表示をする/表示しない
f・7：キーボードからの入力を表示する/しない
f・8：LPに出力する/しない
f・9：画面の文字をプリンタへ出力する。
f・10：LPのラインフィードを行う。

f・6～f・8は1度押すと現在の逆の状態になります。

• ターミナルモードに入った直後はf・6～f・8は次のように設定されています。

f・6　：コントロールコードの表示をしない。
f・7　：キーボードからの入力を表示しない。
f・8　：LPに出力しない。

• GRPH ＋ B の入力で，ターミナルモードからBASICに戻ります。

参照　PC-8001MKII ユーザーズマニュアル

サンプルプログラム

```
term j,2,1,1
```

TIME$　N80 N　予約変数

機能 内蔵のカレンダ時計の時刻を与えます。

書式 1) **TIME$**

2) **TIME$＝"HH : MM : SS"**

文例 **PRINT TIME$**

解説
- TIME$は予約変数であり現在の時刻を持っています。
- TIME$の形式は,

  HH : MM : SS

で示されます。

- TIME$は次のように新たに時刻を設定することもできます。HH は 00〜23まで, MMは 00〜59まで, SSは00〜59までの数です。

　TIME$＝"12：30：00"

参照 DATE$

サンプルプログラム

```
100 ' ｼﾃｲ ｼﾞｶﾝｺﾞ ﾉ ｼﾞｺｸ ｦ ﾓﾄﾒﾙ
110 T$=TIME$
120 PRINT "ｹﾞﾝｻﾞｲ ﾉ ｼﾞｺｸ ﾊ ";T$;"ﾃﾞｽ"
130 INPUT "ｼﾞｶﾝ (0-23):";H
140 INPUT "ﾌﾝ   (0-59):";M
150 INPUT "ﾋﾞｮｳ (0-59):";S
160 HH=VAL(LEFT$(T$,2))
170 MM=VAL(MID$(T$,4,2))
180 SS=VAL(RIGHT$(T$,2))
190 SS=SS+S:MM=MM+M:HH=HH+H
200 FOR I=0 TO 1
210   MM=MM+SS¥60:SS=SS MOD 60
220   HH=HH+MM¥60:MM=MM MOD 60
230   DD=DD+HH¥24:HH=HH MOD 24
240 NEXT I
250 PRINT "ﾓﾄﾒﾙ ｼﾞｺｸ ﾊ"
260 IF DD=1 THEN PRINT "ﾖｸｼﾞﾂ ﾉ";
270 PRINT USING " ##時 ##分 ##秒 ﾃﾞｽ";HH,MM,SS
280 END
run
ｹﾞﾝｻﾞｲ ﾉ ｼﾞｺｸ ﾊ 20:15:54ﾃﾞｽ
ｼﾞｶﾝ (0-23):? 1
ﾌﾝ   (0-59):? 23
ﾋﾞｮｳ (0-59):? 45
ﾓﾄﾒﾙ ｼﾞｺｸ ﾊ
 21時 39分 39秒 ﾃﾞｽ
Ok
```

TRONハイフン/TROFF N80 N 一般コマンド

機能 プログラムの実行状態を追跡します。

書式 **TRON**

TROFF

文例 **TRON**

TROFF

解説 • TRON は，プログラムの誤りを見つけるときなどに実行します。

• TRON を実行すると(ダイレクトモードでもプログラムモードでも実行できます)実行した行番号を表示します。それぞれの行番号は角カッコ([　])に囲まれて表示されます。行番号の表示は TROFF (または NEW)を実行すると停止されます。

サンプルプログラム

```
10 FOR I=1 TO 3
20   PRINT I
30 NEXT I

tron
Ok
run
[10][20] 1
[30][20] 2
[30][20] 3
[30]
Ok
troff
Ok
run
 1
 2
 3
Ok
```

USR　[N_{80}] [N]　特殊関数

機能　**機械語で作られた関数ルーチンを呼び出します。**

書式　**USR〔〈番号〉〕(〈引き数〉)**

文例　**I=USR3(0)**

解説

• USR 関数は，ユーザの作成した機械語関数ルーチンを呼び出します。USR 関数は，呼び出される前に DEF USR で実行開始番地が設定されており，メモリ上に準備されていなければなりません。

• 〈番号〉は 0 ～ 9 までの値で，DEF USR により定義された番号に対応します。〈番号〉が省略された場合は 0 と見なされます。

• 〈引き数〉は数値式あるいは文字式で，その引き数の型は A レジスタに渡されます。詳しくは，付録B機械語プログラムの作り方を参照してください。

注意

N_{80}-BASIC, N_{80} DISK-BASIC において 6000H から 7FFFH 番地に書かれた機械語プログラムを USR 関数で呼び，その値を，直接 CMD, STATUS のついた文の引き数として使うことはできません。付録Bソフトウェア上の一般的注意事項を参照してください。

参照　DEF USR 付録B　機械語プログラムの使い方

サンプルプログラム

```
100 CLEAR 300,&HDFFF
110 DEF USR0=&HE000
120 DEF USR1=&HE010
130 POKE &HE000,&HC9
140 POKE &HE010,&HC9
150 A=USR0(123)
160 PRINT A
170 PRINT USR1(456)

run
 123
 456
Ok
```

VAL　N80　N　文字関数

機能　**文字列の表す数値を与えます。**

書式　**VAL**(<文字列>)

文例　**A=VAL("2")**

解説

• <文字列>を数値に変換して返します。

• <文字列>の最初の文字が+，－，&，.，または数字でなければVALの値は0になります。また，数字以外の文字(ただし，16進数の場合のA～F，&HのH，および指数表現のDとEを除く。)が含まれている場合，それ以降の文字は無視されます。

• 文字列中のスペースは無視します。

参照　STR$

サンプルプログラム

```
100 ' 00:00:00 ｶﾗ ｹﾞﾝｻﾞｲ ﾏﾃﾞﾉ
110 ' ﾋﾞｮｳｽｳ ｦ ﾓﾄﾒﾙ
120 T$=TIME$
130 H=VAL(LEFT$(T$,2))
140 M=VAL(MID$(T$,4,2))
150 S=VAL(RIGHT$(T$,2))
160 SS=H*3600+M*60+S
170 PRINT "00:00:00 ｶﾗ ";T$;" ﾏﾃﾞﾊ";
180 PRINT SS;"秒 ｶﾝ ﾃﾞｽ"
190 END

run
00:00:00 ｶﾗ 10:12:02 ﾏﾃﾞﾊ 36722 秒 ｶﾝ ﾃﾞｽ
Ok
```

VARPTR　N80 N　特殊関数

(機　能)　**変数の格納されているメモリ番地，ファイルに割り当てられているバッファの先頭番地を与えます。**

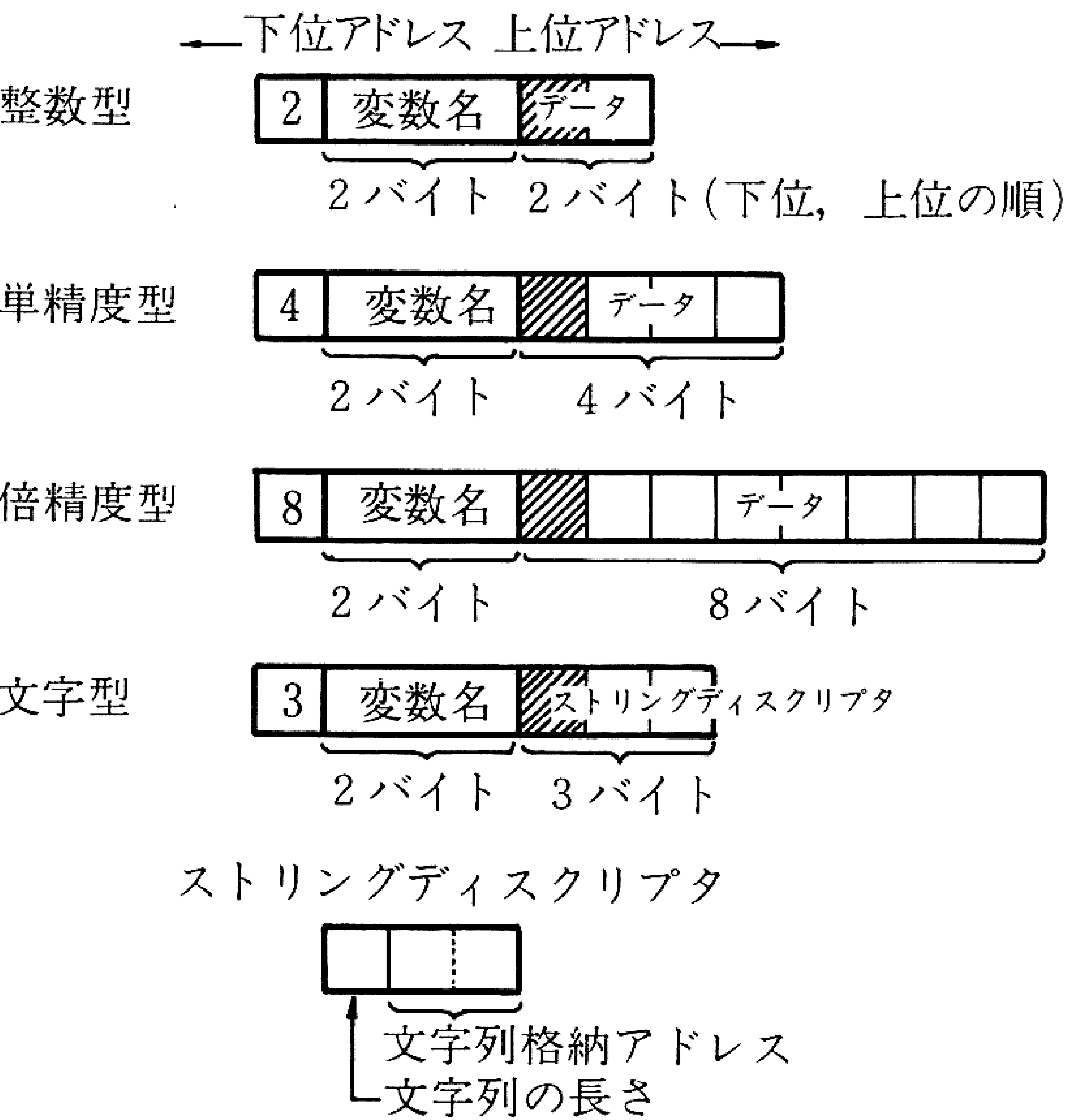
(書　式)　**1）VARPTR(<変数名>)**

2）VARPTR(＃<ファイル番号>)

(文　例)　**1）PRINT HEX$(VARPTR(A))**

2）BA＝VARPTR(＃1)

(解　説)　1) <変数名>で指定した変数のデータが格納されている変数領域のメモリ番地を与えます。このとき指定される変数は値が代入されていなければなりません。



2) 指定した<ファイル番号>に割り当てられている入出力バッファの先頭番地－9を与えます。

(参　照)　付録B機械語プログラムの作り方(VARPTR)

サンプル
プログラム

```
100 '-- (local value) ¥ 2 ヲ モトメル --
110 DEFINT A
120 INPUT "Value";A
130 AD=VARPTR(A)
140 AL=(A/2) MOD 256
150 AH=(A¥2)¥256
160 POKE AD,AL
170 POKE AD+1,AH
180 PRINT A
190 GOTO 120

run
Value? 1245
 622
Value? 653
 326
Value? 9035
 4517
Value?
Break in 120
Ok
```

WAIT　N80 N　特殊ステートメント

機能　**入力ポートをモニタし，その間プログラムの実行を中断します。**

書式　**WAIT〈I/Oアドレス〉,〈式1〉〔,〈式2〉〕**

文例　**WAIT 1, &H22, &H22**

解説

• WAIT は指定した入力ポートのビットパターンが指定した状態になるまでプログラムの実行が中断されます。

• 指定したポートから読み込んだデータと〈式 2〉との XOR の結果と〈式 1〉との AND が取られ，その結果が真(0 以外)になるまでポートのデータを読み続けます。XOR と AND の結果を A として式で表わせば次のようになります。

A =((読み込んだデータ) XOR〈式 2〉) AND〈式 1〉

この結果(この場合 A)が 0 (偽)であれば BASIC はもう一度ポートのデータを読み込み同じ操作を繰り返します。もし結果が 0 でなければ(真)，プログラムの実行は次の文に移ります。

• 〈式 2〉が省略された場合は 0 と見なします。

注意　WAIT は式の指定を誤ると無限ループに入ってしまいます。この場合 MKⅡ をリセットしなければなりません。

サンプルプログラム

```
100 '-- read from key port --
110 PRINT "Type 's' to start."
120 WAIT 4,8,255
130 'start
140 END

run
Type 's' to start.
Ok
s
```

WIDTH　[N80] [N]　画面制御ステートメント

機能　**テキスト画面に表示する文字の行数と桁数を指定します。**

書式　**WIDTH〔〈桁数〉〕〔,〈行数〉〕**

文例　**WIDTH 80, 25**

解説

• テキスト画面に表示する1画面あたりの行数と1行あたりの桁数を指定します。

• 〈桁数〉は80,72,40,36のいずれかで指定します。ただしテキスト画面がカラーモードの状態にあるときは1行あたりに表示する桁数は〈桁数〉－1となります。また，〈桁数〉が省略された場合，現在の〈桁数〉が採用されます。

• 〈桁数〉は20か25のどちらかを指定します。省略された場合，現在の〈桁数〉が採用されます。

• 低解像度グラフィックの解像度はWIDTHによって決定されます。

解像度

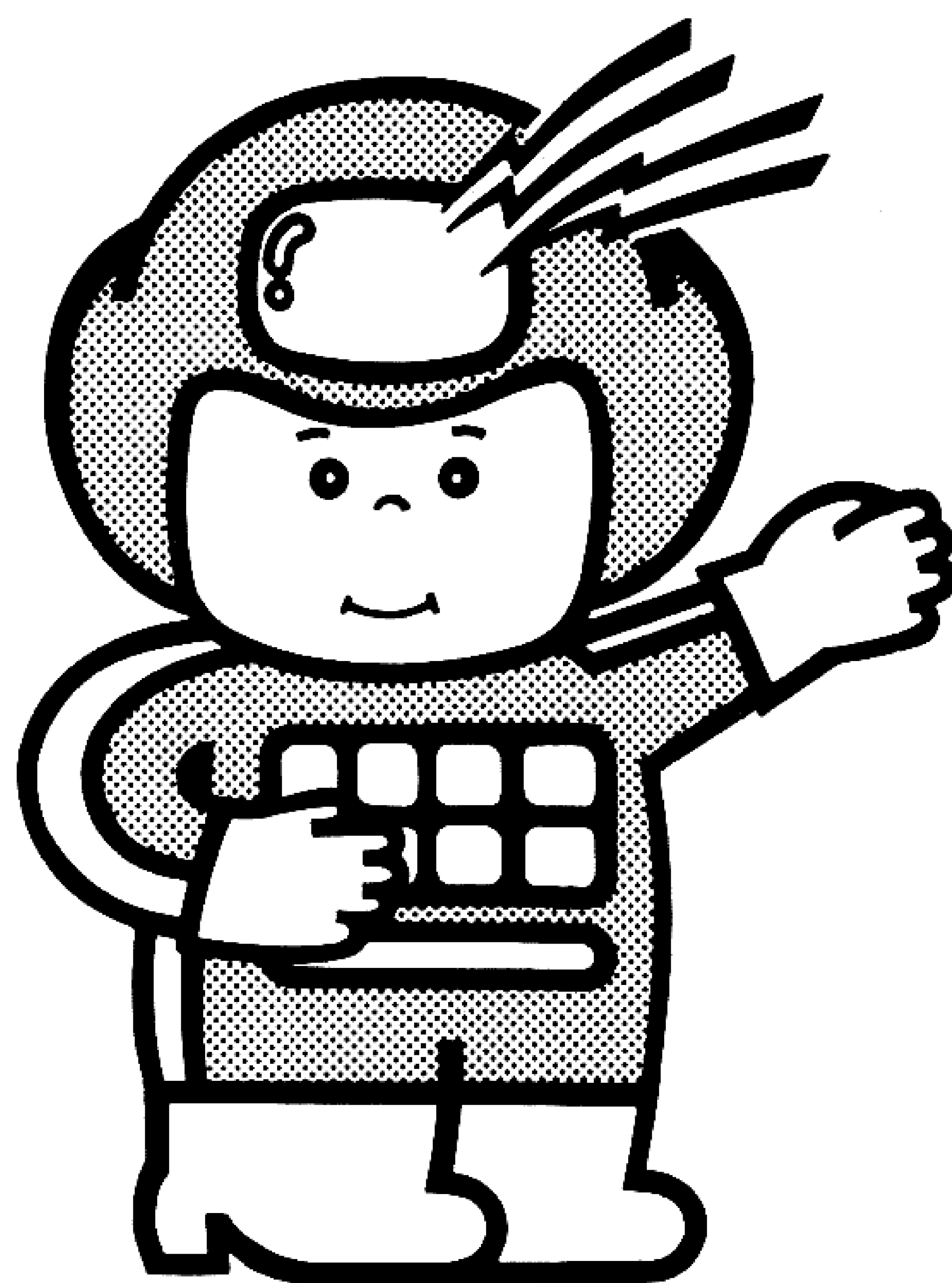
横	〈桁数〉×2	ピクセル
縦	〈行数〉×4	ピクセル

ただし，低解像度グラフィック命令を使用するときは，通常〈行数〉は25を指定します。20を指定するとピクセルとピクセルの間にすきまができます。

サンプルプログラム

```
width 80,25
width ,20
widht 40,20
width 72,25
width 36,20
width 80
```

付　録



A．データファイルの作り方

データファイルには，シーケンシャルファイルとランダムファイルの2種類があります。ここでは入出力の対象としてフロッピィディスクを例にとって，データファイルの作り方を説明します。

A.1　シーケンシャルファイル

シーケンシャルファイルは，ランダムファイルよりも簡単に作ることができますが，データをアクセスする時に，融通性とスピードの点でランダムファイルより制限を受けます。これは，シーケンシャルファイルがデータを順番に記録し，読み出す時も最初から順番に読まなければならないからです。

シーケンシャルファイルをアクセスする際に使用されるステートメントと関数を以下に示します。

OPEN, CLOSE, INPUT＃, LINE INPUT＃, PRINT＃,
PRINT＃ USING, EOF, LOF, LOC, INPUT$(X, 〔＃〕Y)

シーケンシャルファイルを作り，その中のデータをアクセスするために必要な手順は次のようになります。

A.1.1　シーケンシャルファイルの作成

1）OUTPUTモードでファイルをオープンします。

(Sample) **OPEN "TEST" FOR OUTPUT AS ＃1**

2）PRINT＃を使ってファイルにデータを書き込みます。

(Sample) **PRINT＃1, N$; " , " ; P**

3）必要なだけ2）の作業を繰り返した後，ファイルをクローズして終了します。

(Sample) **CLOSE＃1**

例として，キーボードから生徒のテストの点数を入力して，シーケンシャルファイル"TEST"を作るプログラムA—1と実行例を示します。

```
10 'A-1
20 OPEN "TEST" FOR OUTPUT AS #1
30 '
40 INPUT "ﾅﾏｴ";N$
50 IF N$="end" THEN 120
60 INPUT"ｶﾓｸ";K$
70 INPUT"ﾄｸﾃﾝ";P
80 PRINT#1,N$;",";K$;",";P
90 PRINT
100 GOTO 30
110 '
120 CLOSE #1:END

run
ﾅﾏｴ? ﾜﾀﾅﾍﾞ ﾓﾘﾏｻ
ｶﾓｸ? ｺｸｺﾞ
ﾄｸﾃﾝ? 81

ﾅﾏｴ? ｺﾏﾂ ｼｹﾞｷ
ｶﾓｸ? ｺｸｺﾞ
ﾄｸﾃﾝ? 49

ﾅﾏｴ? ｻﾏﾀ ｶﾂﾞﾐ
ｶﾓｸ? ｺｸｺﾞ
ﾄｸﾃﾝ? 100

ﾅﾏｴ? end
Ok
```

シーケンシャルファイルを作成するプログラムにおいて，PRINT＃USINGを使用することによって，書式制御したデータを書き込むこともできます。例えば，

PRINT＃1, USING "＃＃＃＃. ＃＃,"; A, B, C, D

を使うと，特に区切り記号を付けることなく，ディスクに数値データを書き込むことができます。この区切り記号とは，シーケンシャルファイルの中のデータの読み込みにINPUT＃を使用するとき，数値データの場合はコンマ，キャリッジリターン，ラインフィード，空白などを区切りとしてデータを読み込むためのものです。LINE INPUT＃を使用する場合は，キャリッジリターンのみを区切り記号とします。

LOC 関数をシーケンシャルファイルについて使用する場合は，そのファイルが開かれて以降，アクセスされた総セクタ数を値とします。

A.1.2 シーケンシャルファイルからの読み込み

1） INPUT モードでファイルをオープンします。

(Sample) OPEN "TEST" FOR INPUT AS＃1

2) INPUT#, LINE INPUT# 等を使ってファイルからデータを読み込みます。

(Sample) **INPUT#, N$, P**

3) 必要なだけ2)の作業を繰り返した後，ファイルをクローズして終了します。

(Sample) **CLOSE#1**

次にプログラムA－2として，プログラムA－1で作ったファイル"TEST"をアクセスし，50点未満の生徒を全部表示する例を示します。

```
10 'A-2
20 OPEN "TEST" FOR INPUT AS #1
30 '
40 INPUT #1,N$,K$,P
50 IF P>=50 THEN 30
60 PRINT N$,K$,P
70 GOTO 30

run
ｺﾏﾂ ｼｹﾞｷ       ｺｸｺﾞ              49
Input past end in 40
Ok
```

プログラムA－2ではファイル中のデータを最初から順番に読み込んでいきます。そして，ファイルの最後でもう読み込むデータがなくなると，40行で "Input past end" エラーが起こります。これを避けるためには，35行に

```
35 IF EOF(1) THEN 100
```

を挿入してファイルの終りをチェックし

```
100 CLOSE #1
110 END
```

のように処理します。

A.1.3 シーケンシャルファイルへのデータの追加

フロッピィディスク上にシーケンシャルファイルが既に作られていて，その後のデータを追加したい場合には，次のような手順で行います。

1) APPEND モードでファイルをオープンします。

OPEN "TEST" FOR APPEND AS#1

2) PRINT# を使ってファイルに追加したいデータを書き込みます。

Sample PRINT＃1 N$; "," ; P

3) 必要なだけ2)の作業を繰り返した後，ファイルをクローズして終了します。

Sample CLOSE＃1

例として，プログラム A−1 で作ったファイル "TEST" にデータを追加する例をプログラム A−3 として示します。

```
10 'A-3
20 OPEN "TEST" FOR APPEND AS #1
30 '
40 INPUT "ﾅﾏｴ";N$
50 IF N$="end" THEN 120
60 INPUT"ｶﾓｸ";K$
70 INPUT"ﾄｸﾃﾝ";P
80 PRINT#1,N$;",";K$;",";P
90 PRINT
100 GOTO 30
110 '
120 CLOSE #1:END
run
ﾅﾏｴ? ｲﾜﾓﾄ ﾖｳｲﾁ
ｶﾓｸ? ｺｸｺﾞ
ﾄｸﾃﾝ? 30

ﾅﾏｴ? ｻｸﾏ ｵｻﾑ
ｶﾓｸ? ｺｸｺﾞ
ﾄｸﾃﾝ? 95

ﾅﾏｴ? ｶｹﾞﾔﾏ ｹﾝｼﾞ
ｶﾓｸ? ｺｸｺﾞ
ﾄｸﾃﾝ? 50

ﾅﾏｴ? end
Ok
```

このプログラムはプログラム A−1の20行を APPEND モードに替えただけです。従って，このプログラムでデータを追加したファイルに対してプログラム A−2 を実行してチェックすることができます。

```
run"A-2"
ｺﾏﾂ ｼｹﾞｷ          ｺｸｺﾞ          49
ｲﾜﾓﾄ ﾖｳｲﾁ        ｺｸｺﾞ          30
Ok
```

A.1.4 シーケンシャルファイルのデータの修正

シーケンシャルファイルではそのファイルのデータを修正することができません。従って，現在のファイルから読み込んで修正したデータを新しい別のシーケンシャルファイルへ書き込んでいく作業を行なわなければな

りません。以下にプログラムA−4として，プログラムA−1やA−3で作ったファイルの修正を行う例を示します。

```
10 'A-4
20 OPEN "TEST" FOR INPUT AS #1
30 OPEN "test" FOR OUTPUT AS #2
40 '
50 IF EOF(1) THEN 500
60 INPUT #1,N$,K$,P
70 IF FLG=1 THEN 300
80 PRINT N$,K$,P
90 '
100 PRINT "ﾍﾝｺｳｶﾞ ｱﾘﾏｽｶ?";
110 INPUT "(y or n or d or e)";A$
120 A$=LEFT$(A$,1)
130 IF A$="e" THEN FLG=1:GOTO 300
140 IF A$="d" THEN 40
150 IF A$="n" THEN PRINT :GOTO 300
160 IF A$<>"y" THEN 100
170 INPUT "ﾅﾏｴ ";N$
180 INPUT "ｶﾓｸ ";K$
190 INPUT "ﾄｸﾃﾝ";P
200 PRINT
300 '
310 PRINT #2,N$;",";K$;",";P
320 GOTO 40
500 '
510 CLOSE #1
520 CLOSE #2
530 KILL"TEST"
540 NAME"test" AS "TEST"
550 END
```

上のプログラムでは，ファイル“TEST”から読み込んだデータに修正がない場合はそのまま新しいファイル“test”へ書き込みます。もし修正がある場合は，170〜190行で新しいデータを入力し，“test”へ書き込みます。削除する場合には書き込まずに次のデータを“TEST”から読み込みます。修正が終了した場合には，“TEST”から読み込んだデータをそのまま“test”へ書き込む作業を“TEST”のファイルエンドまで繰り返します。

最後に，古いファイル“TEST”を消去し，“test”を“TEST”に名前を変更して終了します。

A.2 ランダムファイル

ランダムファイルを作るには，シーケンシャルファイルの場合より多くの手順が必要となります。

しかし，ランダムファイルでは数値を内部表現に変換して記録するためファイルの占める領域を小さくすることができます。また，レコード単位

で任意のデータのアクセスができるため，シーケンシャルファイルの場合のように空読みする必要がありません。さらに，データの修正を自由に行えるという利点もあります。

ランダムファイルを作ったりアクセスしたりするには次の手順が必要となります。

1）ランダムファイルをオープンします。

(Sample) **OPEN "TESTR" AS#1**

2）ランダムバッファに文字変数を割り当てます。

(Sample) **FIELD #1, 20 AS K$, 2AS P$**

3）データの書き込み，読み込みを行います。

(Sample) **PUT#1 NM%**

GET#1, NM%

ただし，データを書き込む前に，バッファへデータを転送しておかなければなりません。データが文字列の場合はLSET, RSETを使います。

(Sample) **LSET N$=A$**

また，数値である場合には一旦MKI$, MKS$, MKD$を使って，それぞれ整数値，単精度実数値，倍精度実数値を2バイト，4バイト，8バイトの文字に変換してからバッファへ転送します。

(Sample) **LSET P$=MKI$ (P)**

また，GET# によって読み込んだデータが数値である場合には，CVI, CVS, CVD を使って文字列から数値へ変換します。

(Sample) **P=CVI (P$)**

4）ランダムファイルをクローズして作業を終了します。

(Sample) **CLOSE#1**

次に，プログラムA−5として，プログラムA−1〜A−4を生徒番号によってランダムアクセスを行うランダムファイルにした場合の例を示します。

```
100 'A-5
110  INPUT "file name";F$
120  OPEN F$ AS #1
130  FIELD #1,1 AS ID$,20 AS N$,20 AS K$,2 AS P$
140 ' ﾒﾆｭｰ
150  CMD CLS 1
160  PRINT "  1:ﾃﾞｰﾀ ｻｸｾｲ ﾍﾝｺｳ"
170  PRINT "  2:ﾃﾞｰﾀ ｻﾝｼｮｳ"
180  PRINT "  3:ﾃﾝｽｳ ﾆ ﾖﾙ ｹﾝｻｸ"
190  PRINT "  4:ｼｭｳﾘｮｳ"
200  PRINT "999:ﾌｧｲﾙ ﾉ ｲﾆｼｬﾗｲｽﾞ"
210  INPUT "  ｴﾗﾝﾃﾞ ｸﾀﾞｻｲ ";F
220   IF F=999 THEN 750
230  ON F GOTO 280,250,580,860
240 GOTO 140
250 ' ﾃﾞｰﾀ ｻﾝｼｮｳ
260  SF=-1
270  GOTO 380
280 ' ﾃﾞｰﾀ ﾍﾝｺｳ
290  SF=0
300 '
310  GET #1,1
320  MAX=CVI(P$)
330  PRINT "ｾｲﾄ ﾊﾞﾝｺﾞｳ (1-";USING  "###";MAX-1;
340  INPUT " ﾏﾀﾊ 0:ｵﾜﾘ)";NM%
350   IF NM%=0 THEN 140
360  NM%=NM%+1
370  GET #1,NM%
380   IF ID$=CHR$(255) AND SF=0 THEN 470
390   IF ID$=CHR$(255) AND SF THEN 300
400  P%=CVI(P$)
410  PRINT
420  PRINT "ﾅﾏｴ :";N$
430  PRINT "ｶﾓｸ :";K$
440  PRINT "ﾄｸﾃﾝ:";P%
450  PRINT
460   IF SF THEN 300
470 ' ｼﾂﾓﾝ
480  INPUT "ﾅﾏｴ ";A$
490   IF A$<>"" THEN LSET N$=A$
500  INPUT "ｶﾓｸ ";A$
510   IF A$<>"" THEN LSET K$=A$
520  INPUT "ﾄｸﾃﾝ";A$
530   IF A$<>"" THEN LSET P$=MKI$(VAL(A$))
540  PRINT
550  LSET ID$="U"
560  PUT #1,NM%
570 GOTO 280
580 ' ｹﾝｻｸ
590  GET #1,1
600  MAX=CVI(P$)
610  INPUT "ﾅﾝﾃﾝｲｶ ﾃﾞ ｹﾝｻｸ ｼﾏｽｶ ";AK
620  PRINT
630  FOR NM%=2 TO MAX
640   GET #1,NM%
650    IF ID$=CHR$(255) THEN 690
660   IF CVI(P$)>AK THEN 690
670  PRINT USING "### ";NM%-1;
680  PRINT N$;K$;USING " ####";CVI(P$)
690  NEXT NM%
700  PRINT
710  PRINT "ｹﾝｻｸ ｼｭｳﾘｮｳ｡ RETURN ｦ ｵｼﾃ ｸﾀﾞｻｲ｡"
720 '
730  A$=INKEY$:IF A$="" THEN 720
740 GOTO 140
750 ' ｲﾆｼｬﾗｲｽﾞ
```

```
760  INPUT "ｾｲﾄ ﾊﾞﾝｺﾞｳ ﾉ ｻｲﾀﾞｲﾁ ﾊ ";MAX
770  INPUT "ﾌｧｲﾙ ｦ ｲﾆｼｬﾗｲｽﾞ ｼﾏｽｶ (y ﾏﾀﾊ n) ";A$
780   IF A$<>"y" THEN 140
790  FOR NM%=1 TO MAX+1
800   LSET ID$=CHR$(255)
810   PUT #1,NM%
820  NEXT NM%
830  LSET P$=MKI$(MAX+1)
840  PUT #1,1
850 GOTO 140
860 ' ｼｭｳﾘｮｳ
870  CLOSE #1
880 END
```

B. 機械語プログラムの作り方

N_{80}-BASICには，機械語で書かれたプログラムを呼び出す機能として，USR関数が用意されています。ここでは機械語プログラムを作成する際の注意とUSR関数との引き数の受け渡し方について説明します。なお，ここでの説明を正しく理解するためには μPD-780C-1のプログラミングに関する知識を必要とします。

B.1 メモリの配置

機械語ルーチンをモニタにより作成したり，ファイルからロードする際には，そのためのメモリ領域を確保する必要があります。この機械語プログラム用のメモリ領域はBASICの使用する領域と重複してはなりません。さもないと，BASICの動作が異常になったり，逆にBASICの動作により準備した機械語プログラムが破壊されたりします。メモリの確保はCLEAR文によりBASICの使用するメモリの上限を設定することにより行います。指定された上限以上の領域はBASICは利用しなくなりますので機械語プログラム領域として用いることができます（CLEAR文参照）。

B.2 機械語プログラムの準備

確保した領域に機械語プログラムを準備する主な方法としては次の3つがあります。

1. モニタを用いる

MON コマンドにより機械語モニタに移り，モニタの機能を利用してプログラムを書き込む（PC-8001MKIIユーザーズマニュアル参照）。

2. CMD BLOADによりロードする

既にCMD BSAVEによりセーブされている機械語プログラムをロードする。

3. POKEによりプログラムを書き込む。

比較的短いプログラムであれば，DATA文により機械語プログラムをデータの型で表現しておき，それをREAD文で読み出した後POKE文により書き込む。

また機械語プログラムを作成する際には次の点に注意してください。

注　意

1. BASICに制御を戻すためには，プログラムはリターン命令(RET)を用いなければなりません。

2. USR関数で機械語ルーチンに引き渡された引き数が文字列であった場合には，(DE)レジスタペアに次のような3バイトからなるストリング・ディスクリプタと呼ばれる領域の，先頭のアドレスが入ります。

   (a) ストリング・ディスクリプタの最初のバイトはその文字列の長さ（0から255）を保持します。

   (b) 2番目のバイトは文字列の置かれている先頭アドレスの下位8ビットを保持します。

   (c) 最後のバイトは，文字列の置かれている先頭アドレスの上位8ビットを保持します。

   機械語ルーチンではこの3バイトのストリング・ディスクリプタを書き換えてはなりません。またストリング・ディスクリプタによって指されるストリングの内容は変更することはできますが，その長さを変更してはなりません。

3. 機械語ルーチンからBASICに戻る時には，呼び出された時とスタックポインタ(SP)の値が等しくなっていなければ

なりません。

4. 機械語ルーチンが呼び出された時，スタック領域としては 8 レベル(16バイト)分だけ利用できるように設定されています。もし，機械語ルーチン内でより大きなスタックを必要とする場合には，ルーチン内で独自にスタックを用意しなければなりません。また，この時，BASIC に戻る時にスタックポインタを戻すことを忘れてはなりません。

B.3 USR関数の呼び出し

USR 関数は次のような書式に使います。

(書　式) USR 〔〈**番号**〉〕〔〈**引き数**〉〕

ただし〈番号〉は 0 から 9 までの数値で，引き数は任意の数値式または文字式です。USR 関数は書式上〈引き数〉を必要としますので，呼び出されたルーチンが引き数を必要としない場合でも，ダミーの引き数を与える必要があります。

USR 関数が呼び出され，対応する機械語ルーチンに制御が移った時，A レジスタは，渡された引き数の型を示す値を持っています。その値と意味は次のようになっています。

A の値	引き数の型
2	2 バイトの整数(2の補数表現)
3	文字列型
4	単精度型
8	倍精度型

引き数が文字列の場合には〔DE〕レジスタペアは，3 バイトから成るストリング・ディスクリプタを指しています(前節参照)。

引き数が数値の場合には，〔HL〕レジスタに浮動小数点アキュームレータ (FAC) と呼ばれる 8 バイトの領域の 5 バイト目のアドレスが入っています。実際の引き数は，この FAC の内に各型に応じて以下のように格納されて渡されます。

• 整数型のとき

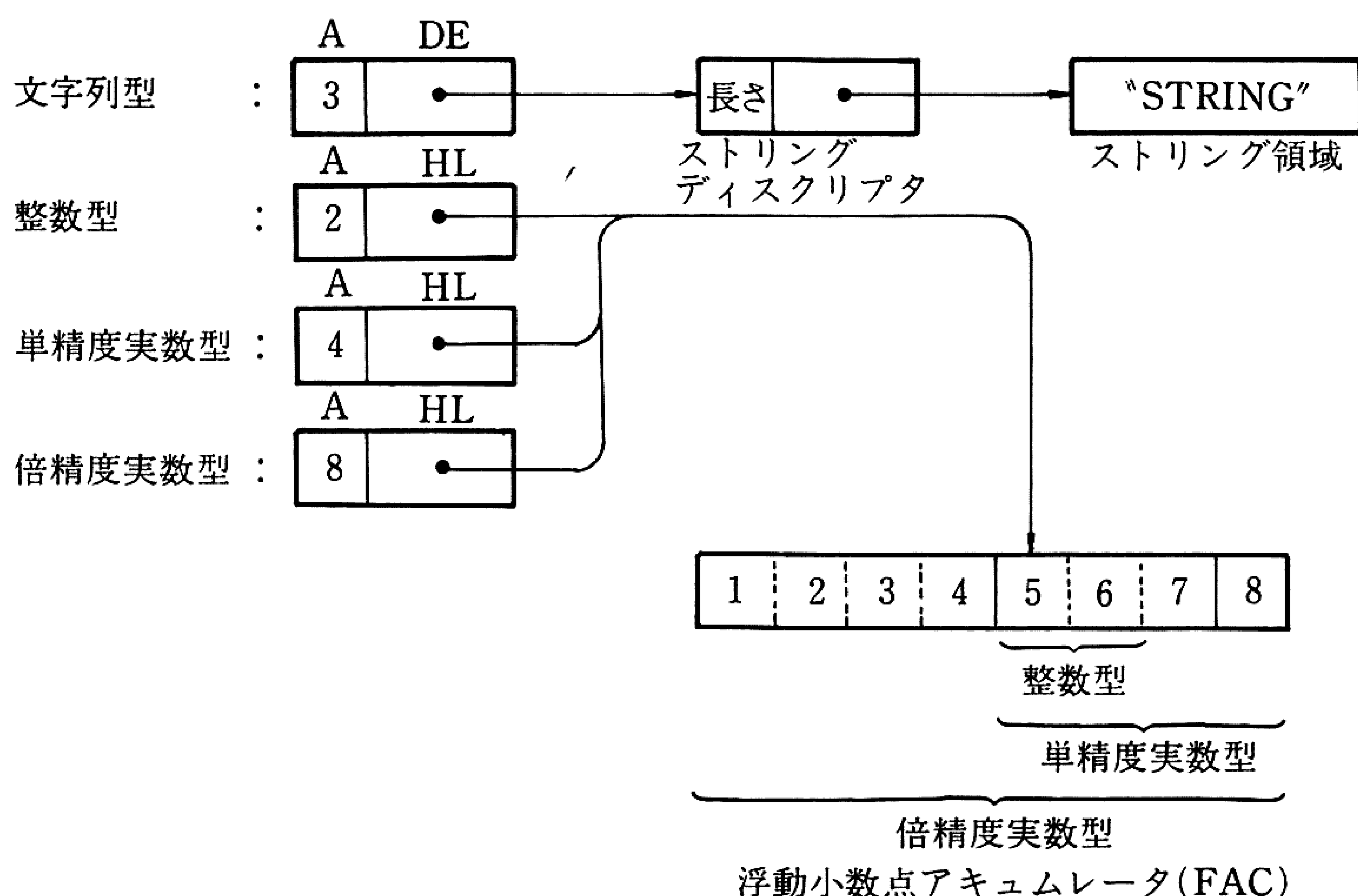
FACの5バイト目(〔HL〕レジスタペアが指しているところ)に引き数の下位8ビット，6バイト目に上位8ビットが入ります。

• 単精度実数型のとき

FACの8バイト目は指数部になっており，(指数—128)の値が入ります。小数点は仮数部の最上位ビットの左にあると想定します。7バイト目は仮数部の最高部7ビットを保持します。このバイトの最上位ビットは付号を示し，0で正，1で負を表します。6バイト目と5バイト目は，それぞれ仮数部の中部と最低部の8ビットを保持します。

• 倍精度実数値のとき

5バイト目から8バイト目までは単精度実数型と同じように指数部と仮数部の上位3バイトが入ります。1バイト目から4バイト目には仮数部の下位4バイトが入ります。



USR関数がBASICに値を返すには関数値をFACに設定してやることで行えます。通常返す値は，引き数として渡した値と同じ型になります。しかし，BASICの持っているMAKINTというルーチンを呼べば，〔HL〕レジスタの整数値を関数値として戻すように強制することができます。

MAKINT ルーチンは機械語プログラムから戻る時に次の様に用います。

CALL 23H ; MAKINTルーチンを呼ぶ

RET ; 機械語プログラムからBASICに戻る

サンプルプログラム

```
100 CLEAR 300,&HBFFF
110 DEFINT A-Y:DEFSTR Z:DIM Z$(1)
120 DEF USR=&HC000:GOSUB 420
130 Z(0)="〜ヽ":Z(1)="ヽ∪ノ"
140 ZS="┌◆┐":Z="|":ZC="   "
150 CONSOLE 0,25,0,1:WIDTH 80,25
160 COLOR 6,32,0
170 PRINT CHR$(12)
180 N=0:K=38:L=39:A=23:B=24:S=1:W=0
190 LOCATE K,B:PRINT ZS;
200 LOCATE L,A:PRINT Z;
210 H=0
220 LOCATE H,N:PRINT Z(H MOD 2);
230 IF W THEN GOSUB 270:GOTO 250
240 IF INP(9)=&HBF THEN W=1 ELSE GOSUB 410
250 LOCATE H,N:PRINT ZC;
260 H=H+1:IF H>76 THEN 210 ELSE 220
270 A=A-1:IF A>=N THEN 310
280 A=23:W=0:LOCATE L,N:PRINT" ";
290 LOCATE L,A:PRINT Z;
300 RETURN
310 AD=&HF328+A*120:D=USR(AD):IF D THEN 330
320 RETURN
330 FOR I=0 TO 5
340 LOCATE L,N:PRINT"X";:BEEP 1:GOSUB 410
350 LOCATE L,N:PRINT" ";:BEEP 0:GOSUB 410
360 LOCATE L-1,N:PRINT"   ";
370 LOCATE L,N+1:PRINT" ";
380 NEXT I:A=23:H=77:W=0
390 LOCATE L,A:PRINT Z;
400 RETURN
410 FOR J=0 TO 10:NEXT J:RETURN
420 FOR AD=&HC000 TO &HC027
430 READ DA:POKE AD,DA
440 NEXT AD
450 RETURN
460 DATA &HFE,2           :'      CP      2
470 DATA &HC0             :'      RET     Z
480 DATA &H5E             :'      LD      E,(HL)
490 DATA &H23             :'      INC     HL
500 DATA &H56             :'      LD      D,(HL)
510 DATA &HE5             :'      PUSH    HL
520 DATA &HEB             :'      EX      DE,HL
530 DATA &H7E             :'      LD      A,(HL)
540 DATA &HFE,&H76        :'      CP      '∪'
550 DATA &HCA,&H1F,&HC0:'         JP      Z,HIT
560 DATA &H36,&H96        :'      LD      (HL),'|'
570 DATA &H11,&H78,0      :'      LD      DE,120
580 DATA &H19             :'      ADD     HL,DE
590 DATA &H36,&H20        :'      LD      (HL),' '
600 DATA &HE1             :'      POP     HL
610 DATA &H36,0           :'      LD      (HL),0
620 DATA &H2B             :'      DEC     HL
630 DATA &H36,0           :'      LD      (HL),0
640 DATA &H3E,2           :'      LD      A,2
650 DATA &HC9             :'      RET
660 DATA &HE1             :'  HIT:POP     HL
670 DATA &H36,&HFF        :'      LD      (HL),&HFF
680 DATA &H2B             :'      DEC     HL
```

```
690 DATA &H36,&HFF      :'      LD      (HL),&HFF
700 DATA &H3E,2         :'      LD      A,2
710 DATA &HC9           :'      RET
```

USR関数を使って，UFOを打ち落とすゲームを作ってみました。スペースを押すとミサイルが発射されます。

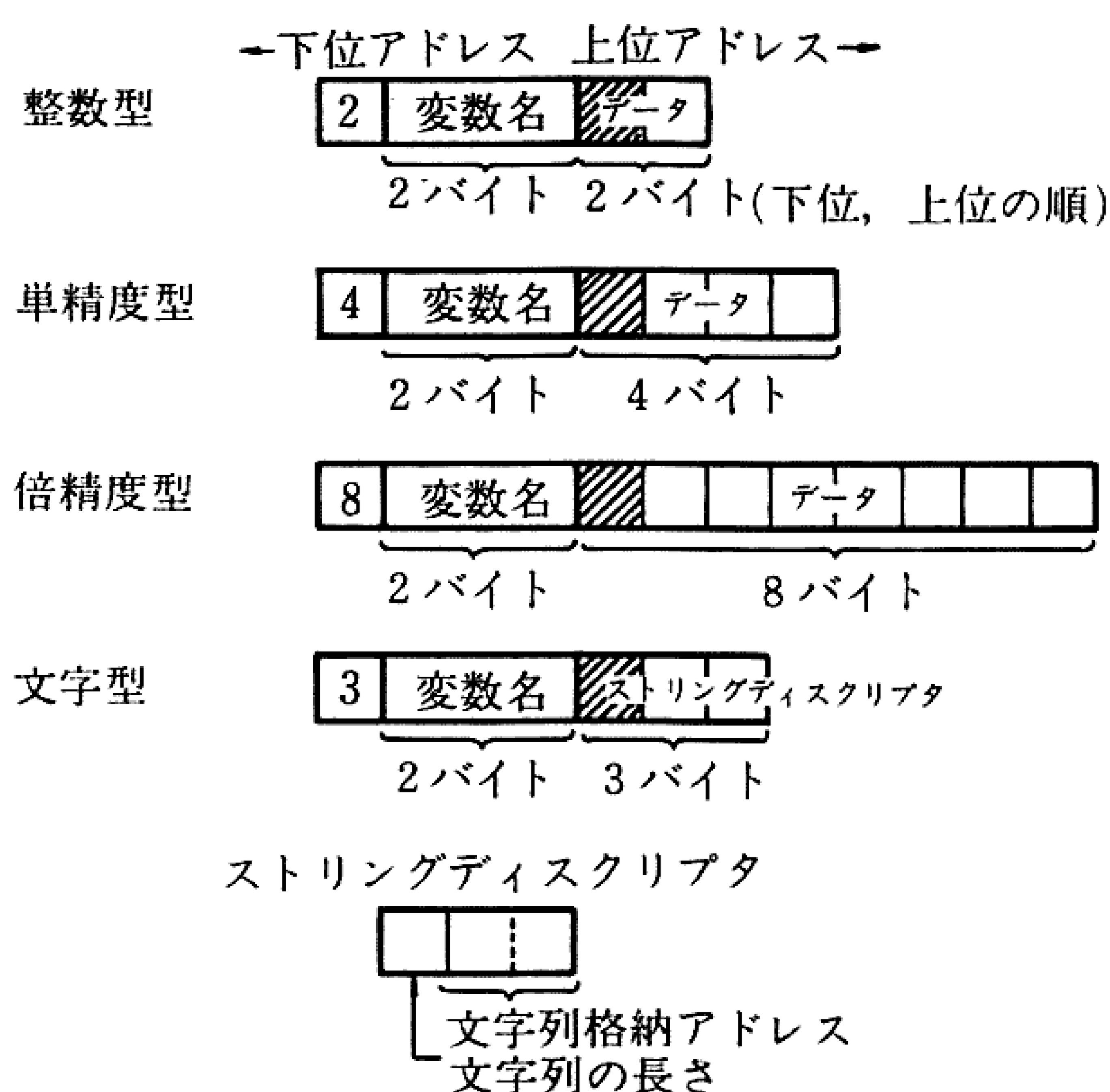
ここでは，機械語のプログラムは，DATA文で用意しておきPOKEでC000H番地以降に書き込んでいます。

機械語ルーチンは，ミサイルの表示と，ミサイルがUFOに当たったのかどうかの判定を行います。行番号310でUSR関数の引き数として，次にミサイルを表示する位置のVRAMのアドレスを渡しています。機械語ルーチンでは，そのアドレスUFOがいるかどうか調べ，もしいれば−1，いなければミサイルを移動して0を返します。

B.4 VARPTR

VARPTR関数は指定された変数，バッファの置かれている番地を返します。

VARPTR関数の引き数として数値変数名が与えられた時には，その変数の値の最下位アドレスを返します。上位バイトは下図に示すようにこの番地以降に入っています。また，文字変数名が指定された場合にはストリング・ディスクリプタの先頭番地が返されます。



変数として配列要素が指定された場合には，その要素の置かれている番地が返されます。

引き数としてバッファ番号が与えられた場合には，そのバッファ領域の番地を返します。この領域の最初の9バイトはBASICの作業用の領域となっており，ファイルとの入出力に用いられる256バイトの領域は，VARPTR関数の値に9を加えた番地から始まります。

C．エラーコード表

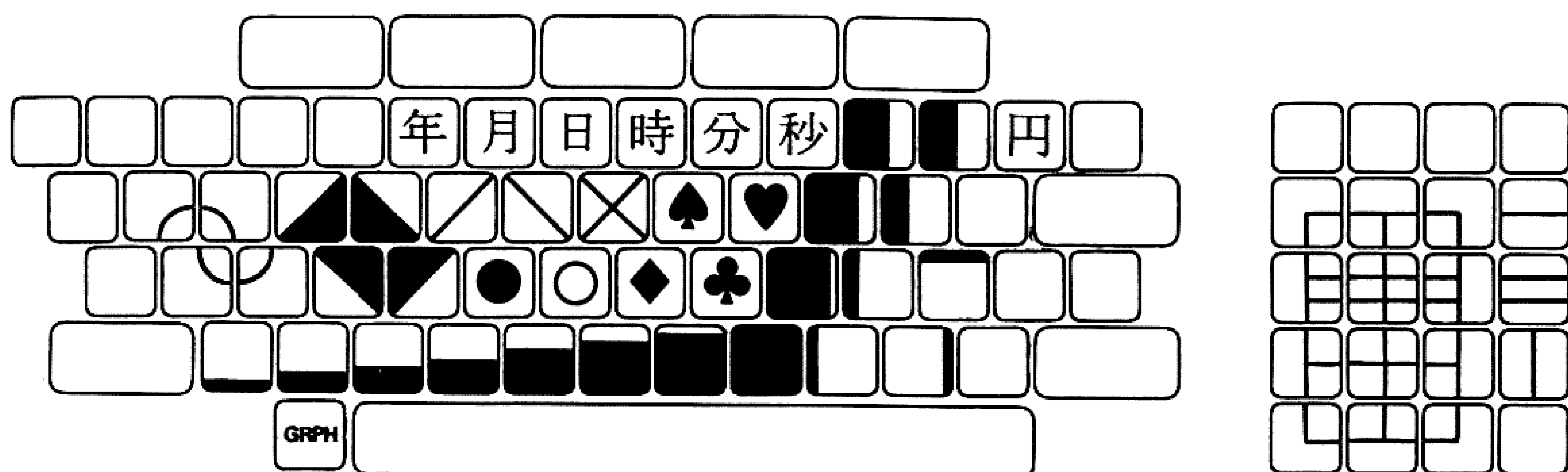
エラーメッセージ	コード	意　　味
Bad allocation table	64	フロッピィディスク内部のFATが壊れている。
Bad drive number	65	ドライブ指定が誤っている。システムにないドライブを指定した。
Bad File Data	25	ファイル上にあるデータの形式がまちがっている。
Bad file name	62	ファイル名の指定がまちがっている。
Bad file number	52	オープンしていないファイルや起動時に指定していないファイルを参照した。
Bad file mode	54	シーケンシャルでオープンしたファイルに対してランダムアクセスをしようとした。またはその逆。
Bad surface track/sector	66	サーフェストラック，セクタ番号の指定が誤っている。(DSKO$, DSKI$)
Can't continue	17	CONT命令によって実行を再開することができない。(ポインタの値が壊されている)。
Communications Buffer Overflow	27	周辺機器との入出力のためのバッファがオーバーフローした。

エラーメッセージ	コード	意　　味
Deleted record	67	消去済のレコードをアクセスしようとした。
Direct Statement in file	63	アスキー形式のファイルを読み込む際にダイレクトステートメントが存在した。
Disk already mounted	59	mountされているディスクに対してmountを行った。
Disk BASIC Feature	26	ディスクが接続されていないとき，(N80) DISK-BASICの命令を実行した。
Disk full	60	ディスク上に書き込むスペースがないのに書き込もうとした。
Disk not mounted	56	mountされていないディスクに対してアクセスした。
Disk I/O Error	57	ディスクとの入出力中にエラーが発生した。致命的なエラーであり回復させることはできない。
Disk offline	73	入出力の可能な状態でないディスクをアクセスしようとした。
Division by zero	11	0による割り算が実行されて，その結果の値がオーバーフローした。
FIELD overflow	50	FIELD文において256バイト以上の大きさの領域を指定した。
File already exists	58	NAME文によって変更しようとしたファイル名が既に存在している。
File already open	55	既にオープンされているファイルに対してOPEN，KILLなどを実行した。
File not found	53	LOAD，SAVE,KILLなどで現在のディスクに存在しないファイルを指定した。
File not OPEN	71	オープンされていないファイルを参照しようとした。
File write protected	72	書き込み禁止属性が付けられているファイルに書き込もうとした。
Illegal direct	12	ダイレクトモードで使用できない文を実行しようとした。
Illegal function call	5	ステートメントおよび関数において，機能の呼び方が誤っている。

エラーメッセージ	コード	意　　味
Input past end	61	ファイル中の全てのデータを読み尽した後に，さらに入力文が実行された。
Internal error	51	BASIC内部にエラーが生じた。
Line buffer overflow	23	1行で入力できる文字の範囲を越えて入力が行われた。
Missing operand	22	ステートメント中必要なパラメータが指定されていない。
NEXT without FOR	1	FOR～NEXTが正しく対応していない。（NEXTが多い。）
No RESUME	19	エラー処理ルーチンの中にRESUME文がなく，プログラムの実行が継続できない。
Out of data	4	読むべきデータがないのにREADが実行された。
Out of memory	7	メモリ容量が足りなくなった。（プログラムが大きすぎる。またはスタックを消費し尽した。）
Out of string space	14	文字列を格納するメモリ領域がなくなった。
Overflow	6	演算結果や入力された数値が，許される範囲を越えた。
Port not initialized	28	インタフェース用のLSIの機能設定がなされていない。
Position not on Screen	24	指定したカーソル位置などが，画面の範囲外になっている。
Redimensioned array	10	配列またはユーザ関数を二重定義しようとした。
Rename across disks	68	NAME文において，異なるドライブ間でリネームしようとした。
RESUME without error	20	エラー処理ルーチンに制御を移さなかったのにRESUME文がある。
RETURN without GOSUB	3	GOSUB～RETURNが正しく対応していない。（RETURNが多い。）
Sequential after PUT	69	PUT文実行後，シーケンシャルファイルをアクセスしようとした。

エラーメッセージ	コード	意　　味
Sequential I/O Only	70	シーケンシャル入出力以外は行ってはならない。
String formula too complex	16	文字式が複雑すぎる。(カッコのネスティングが深すぎるなど。)
String too long	15	1つの文字変数内の文字数が225文字を越えている。
Subscript out of range	9	配列変数の添字の値が規定の範囲から外れている。
Syntax error	2	文の記述が文法通りになっていない。
Tape read ERROR	29	テープからの読み込み時にエラーが発生した。
Type mismatch	13	式の左辺・右辺,関数の引数などにおいて変数の型が一致していない。
Undefined line number	8	飛び先として必要とされる行番号(GOTO, GOSUBなどの飛び先)が存在しない。
Undefined user function	18	DEFFN文によって定義されていないユーザ関数を引用しようとした。
Unprintable error	未定義	メッセージの定義されていないエラー。

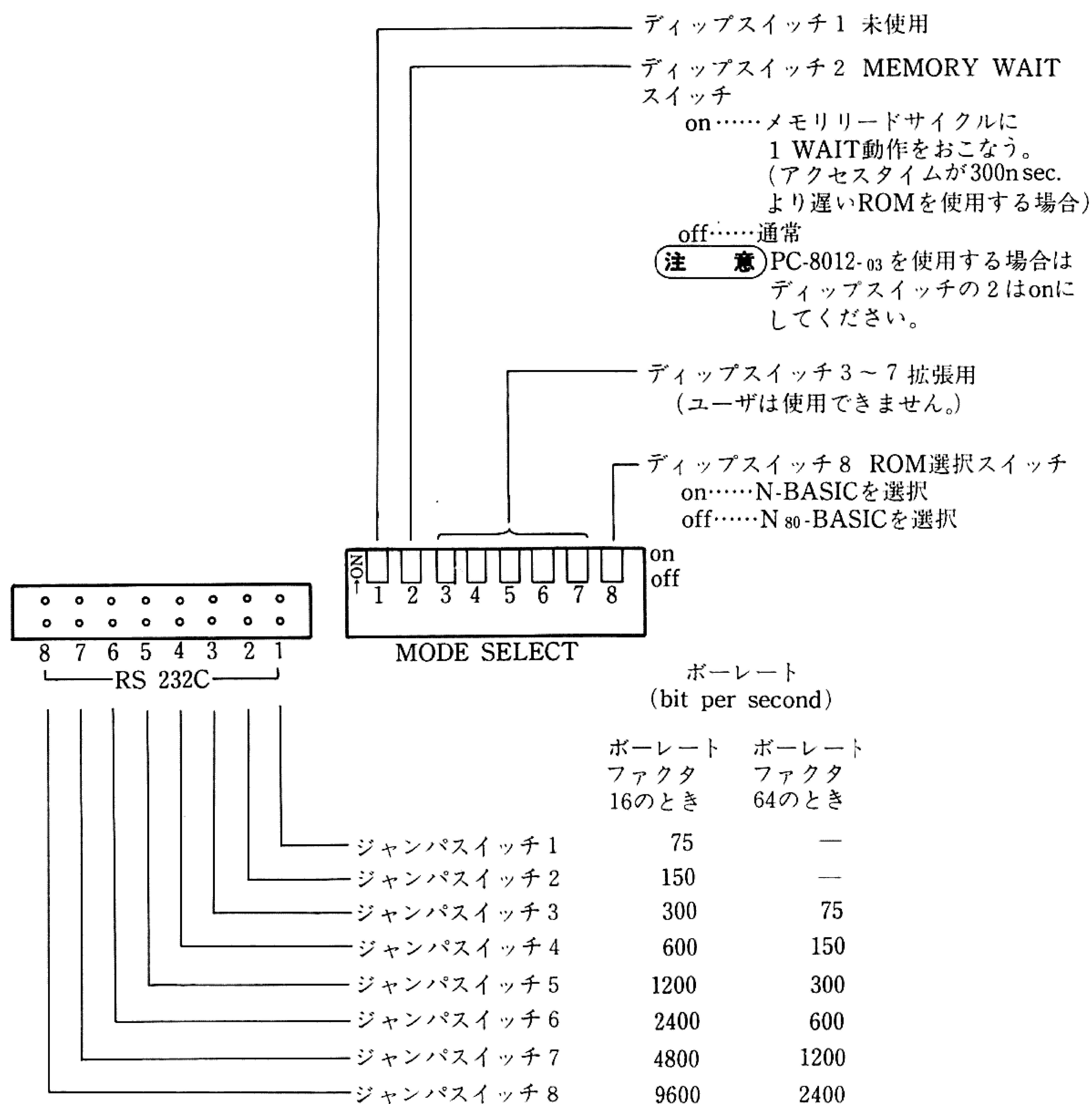
D. グラフィックシンボル



E. キャラクタコード

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		D_E		0	@	P		p	▁	┴		ー	タ	ミ	═	╳
1	S_H	D_1	!	1	A	Q	a	q	▂	┬	。	ア	チ	ム	╞	円
2	S_X	D_2	"	2	B	R	b	r	▃	┤	「	イ	ツ	メ	╪	年
3	E_X	D_3	#	3	C	S	c	s	▄	├	」	ウ	テ	モ	╡	月
4	E_T	D_4	$	4	D	T	d	t	▅	▔	、	エ	ト	ヤ	◢	日
5	E_Q	N_K	%	5	E	U	e	u	▆	━	・	オ	ナ	ユ	◣	時
6	A_K	S_N	&	6	F	V	f	v	▇	┃	ヲ	カ	ニ	ヨ	◥	分
7	B_L	E_B	'	7	G	W	g	w	█	▕	ァ	キ	ヌ	ラ	◤	秒
8	B_S	C_N	(	8	H	X	h	x	▏	┌	ィ	ク	ネ	リ	♠	
9	H_T	E_M	)	9	I	Y	i	y	▎	┐	ゥ	ケ	ノ	ル	♥	
A	L_F	S_B	*	:	J	Z	j	z	▍	└	ェ	コ	ハ	レ	♦	
B	H_M	E_C	+	;	K	[	k	{	▌	┘	ォ	サ	ヒ	ロ	♣	
C	C_L	→	,	<	L	¥	l	¦	▋	╭	ャ	シ	フ	ワ	●	
D	C_R	←	—	=	M	]	m	}	▊	╮	ュ	ス	ヘ	ン	○	
E	S_O	↑	.	>	N	^	n	~	▉	╰	ョ	セ	ホ	゛	╱	
F	S_I	↓	/	?	O	_	o		┼	╯	ッ	ソ	マ	゜	╲	

F．ジャンパスイッチとディップスイッチの使い方



	ボーレートファクタ16のとき	ボーレートファクタ64のとき
ジャンパスイッチ1	75	—
ジャンパスイッチ2	150	—
ジャンパスイッチ3	300	75
ジャンパスイッチ4	600	150
ジャンパスイッチ5	1200	300
ジャンパスイッチ6	2400	600
ジャンパスイッチ7	4800	1200
ジャンパスイッチ8	9600	2400

(注　意) ボートレート切り換え用のジャンパンスイッチはPC-8801のものとは逆にならんでいるので注意してください。

G．演算順位

優先順位	演　算
1	カッコで囲まれたもの
2	関数
3	指数(べき乗) ^
4	負号(－)
5	＊，／
6	¥
7	MOD
8	＋，－
9	関係演算子(＜，＞，＝など)
10	NOT
11	AND
12	OR
13	XOR
14	IMP
15	EQV

H．コントロールコード

キー入力	動　作
CTRL＋B	カーソルを１項目ごと左へ移動させます。
CTRL＋E	カーソルの位置から，その行の終りをまっ消します。
CTRL＋G	ブザーを鳴らします。
CTRL＋H	１文字を削除します。INS DELと同じ働きです。
CTRL＋I	タブ位置までカーソルを移動させます。タブ位置は８桁ごとに設定されています。TABキーと同じ働きです。
CTRL＋J	ラインフィード(LFコード)を挿入します。
CTRL＋K	カーソルをホームポジション(左上スミ)に移動させます。SHIFT＋HOME CLRと同じです。
CTRL＋L	テキスト画面をクリアします。HOME CLRと同じです。
CTRL＋M	RETURN と同じです。
CTRL＋N	カーソルを１項目ごとに右へ移動させます。
CTRL＋R	スペースを挿入します。SHIFT＋INS DELと同じです。
ESC	実行を一時中断します。
⇆	カーソルを１つ右へ移動させます。
SHIFT＋⇆	カーソルを１つ左へ移動させます。
⇅	カーソルを上の行へ移動させます。
SHIFT ⇅	カーソルを下の行へ移動させます。

I．漢字JISコード表

半角文字一覧表

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0020		!	"	#	$	%	&	'	(	)	*	+	,	－	.	/
0030	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
0040	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
0050	P	Q	R	S	T	U	V	W	X	Y	Z	[	¥	]	^	—
0060		a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
0070	p	q	r	s	t	u	v	w	x	y	z	{	¦	}	~	
0080		。	「	」	、	・	を	ぁ	ぃ	ぅ	ぇ	ぉ	ゃ	ゅ	ょ	っ
0090	ー	あ	い	う	え	お	か	き	く	け	こ	さ	し	す	せ	そ
00A0		。	「	」		・	ヲ	ァ	ィ	ゥ	ェ	ォ	ャ	ュ	ョ	ッ
00B0	ー	ア	イ	ウ	エ	オ	カ	キ	ク	ケ	コ	サ	シ	ス	セ	ソ
00C0	タ	チ	ツ	テ	ト	ナ	ニ	ヌ	ネ	ノ	ハ	ヒ	フ	ヘ	ホ	マ
00D0	ミ	ム	メ	モ	ヤ	ユ	ヨ	ラ	リ	ル	レ	ロ	ワ	ン	゛	゜
00E0	た	ち	つ	て	と	な	に	ぬ	ね	の	は	ひ	ふ	へ	ほ	ま
00F0	み	む	め	も	や	ゆ	よ	ら	り	る	れ	ろ	わ	ん	゛	゜
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

1/4角文字一覧表

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0100		SH	SX	EX	ET	EQ	AK	BL	BS	HT	LF	HM	CL	CR	SO	SI
0110	DE	D1	D2	D3	D4	NK	SN	EB	CN	EM	SB	EC	→	←	↑	↓
0120		!	″	#	$	%	&	'	(	)	*	+	,	−	.	/
0130	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
0140	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
0150	P	Q	R	S	T	U	V	W	X	Y	Z	[	¥	]	∧	—
0160		a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
0170	p	q	r	s	t	u	v	w	x	y	z	{	¦	}	~	
0180																+
0190													╭	╮	╰	╯
01A0		。	「	」	、	・	ヲ	ァ	ィ	ゥ	ェ	ォ	ャ	ュ	ョ	ッ
01B0	ー	ア	イ	ウ	エ	オ	カ	キ	ク	ケ	コ	サ	シ	ス	セ	ソ
01C0	タ	チ	ツ	テ	ト	ナ	ニ	ヌ	ネ	ノ	ハ	ヒ	フ	ヘ	ホ	マ
01D0	ミ	ム	メ	モ	ヤ	ユ	ヨ	ラ	リ	ル	レ	ロ	ワ	ン	゛	゜
01E0	＝	╞	╪	╡					♠	♥	♦	♣	●	○	／	＼
01F0	×	円	年	月	日	時	分	秒								
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
記号	2120			、	。	，	．	・	：	；	？	！	゛	゜	´	｀	¨
	2130	＾	￣	＿	ヽ	ヾ	ゝ	ゞ	〃	仝	々	〆	〇	ー	―	‐	／
	2140	＼	～	‖	｜	…	‥	‘	’	“	”	（	）	〔	〕	［	］
	2150	｛	｝	〈	〉	《	》	「	」	『	』	【	】	＋	－	±	×
	2160	÷	＝	≠	＜	＞	≦	≧	∞	∴	♂	♀	°	′	″	℃	￥
	2170	＄	¢	£	％	＃	＆	＊	＠	§	☆	★	○	●	◎	◇	
	2220		◆	□	■	△	▲	▽	▼	※	〒	→	←	↑	↓	〓	
英・数字	2330	0	1	2	3	4	5	6	7	8	9						
	2340		A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
	2350	P	Q	R	S	T	U	V	W	X	Y	Z					
	2360		a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
	2370	p	q	r	s	t	u	v	w	x	y	z					
ひらがな	2420		ぁ	あ	ぃ	い	ぅ	う	ぇ	え	ぉ	お	か	が	き	ぎ	く
	2430	ぐ	け	げ	こ	ご	さ	ざ	し	じ	す	ず	せ	ぜ	そ	ぞ	た
	2440	だ	ち	ぢ	っ	つ	づ	て	で	と	ど	な	に	ぬ	ね	の	は
	2450	ば	ぱ	ひ	び	ぴ	ふ	ぶ	ぷ	へ	べ	ぺ	ほ	ぼ	ぽ	ま	み
	2460	む	め	も	ゃ	や	ゅ	ゆ	ょ	よ	ら	り	る	れ	ろ	ゎ	わ
	2470	ゐ	ゑ	を	ん												
カタカナ	2520		ァ	ア	ィ	イ	ゥ	ウ	ェ	エ	ォ	オ	カ	ガ	キ	ギ	ク
	2530	グ	ケ	ゲ	コ	ゴ	サ	ザ	シ	ジ	ス	ズ	セ	ゼ	ソ	ゾ	タ
	2540	ダ	チ	ヂ	ッ	ツ	ヅ	テ	デ	ト	ド	ナ	ニ	ヌ	ネ	ノ	ハ
	2550	バ	パ	ヒ	ビ	ピ	フ	ブ	プ	ヘ	ベ	ペ	ホ	ボ	ポ	マ	ミ
	2560	ム	メ	モ	ャ	ヤ	ュ	ユ	ョ	ヨ	ラ	リ	ル	レ	ロ	ヮ	ワ
	2570	ヰ	ヱ	ヲ	ン	ヴ	ヵ	ヶ									
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

※コードは16進で表現されています。
例えば〝B〟のコードは2340＋2＝2342となります。

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ギリシア文字	2620		Α	Β	Γ	Δ	Ε	Ζ	Η	Θ	Ι	Κ	Λ	Μ	Ν	Ξ	Ο
	2630	Π	Ρ	Σ	Τ	Υ	Φ	Χ	Ψ	Ω							
	2640		α	β	γ	δ	ε	ζ	η	θ	ι	κ	λ	μ	ν	ξ	ο
	2650	π	ρ	σ	τ	υ	φ	χ	ψ	ω							
ロシア文字	2720		А	Б	В	Г	Д	Е	Ё	Ж	З	И	Й	К	Л	М	Н
	2730	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э
	2740	Ю	Я														
	2750		а	б	в	г	д	е	ё	ж	з	и	й	к	л	м	н
	2760	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э
	2770	ю	я														
ア	3020		亜	唖	娃	阿	哀	愛	挨	姶	逢	葵	茜	穐	悪	握	渥
	3030	旭	葦	芦	鯵	梓	圧	斡	扱	宛	姐	虻	飴	絢	綾	鮎	或
	3040	粟	袷	安	庵	按	暗	案	闇	鞍	杏						
イ	3040											以	伊	位	依	偉	囲
	3050	夷	委	威	尉	惟	意	慰	易	椅	為	畏	異	移	維	緯	胃
	3060	萎	衣	謂	違	遺	医	井	亥	域	育	郁	磯	一	壱	溢	逸
	3070	稲	茨	芋	鰯	允	印	咽	員	因	姻	引	飲	淫	胤	蔭	
	3120		院	陰	隠	韻	吋										
ウ	3120							右	宇	烏	羽	迂	雨	卯	鵜	窺	丑
	3130	碓	臼	渦	嘘	唄	欝	蔚	鰻	姥	厩	浦	瓜	閏	噂	云	運
	3140	雲															
エ	3140		荏	餌	叡	営	嬰	影	映	曳	栄	永	泳	洩	瑛	盈	穎
	3150	頴	英	衛	詠	鋭	液	疫	益	駅	悦	謁	越	閲	榎	厭	円
	3160	園	堰	奄	宴	延	怨	掩	援	沿	演	炎	焔	煙	燕	猿	縁
	3170	艶	苑	薗	遠	鉛	鴛	塩									
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
オ	3170								於	汚	甥	凹	央	奥	往	応	
	3220		押	旺	横	欧	殴	王	翁	襖	鴬	鴎	黄	岡	沖	荻	億
	3230	屋	憶	臆	桶	牡	乙	俺	卸	恩	温	穏	音				
カ	3230													下	化	仮	何
	3240	伽	価	佳	加	可	嘉	夏	嫁	家	寡	科	暇	果	架	歌	河
	3250	火	珂	禍	禾	稼	箇	花	苛	茄	荷	華	菓	蝦	課	嘩	貨
	3260	迦	過	霞	蚊	俄	峨	我	牙	画	臥	芽	蛾	賀	雅	餓	駕
	3270	介	会	解	回	塊	壊	廻	快	怪	悔	恢	懐	戒	拐	改	
	3320		魁	晦	械	海	灰	界	皆	絵	芥	蟹	開	階	貝	凱	劾
	3330	外	咳	害	崖	慨	概	涯	碍	蓋	街	該	鎧	骸	浬	馨	蛙
	3340	垣	柿	蠣	鈎	劃	嚇	各	廓	拡	攪	格	核	殻	獲	確	穫
	3350	覚	角	赫	較	郭	閣	隔	革	学	岳	楽	額	顎	掛	笠	樫
	3360	橿	梶	鰍	潟	割	喝	恰	括	活	渇	滑	葛	褐	轄	且	鰹
	3370	叶	椛	樺	鞄	株	兜	竈	蒲	釜	鎌	噛	鴨	栢	茅	萱	
	3420		粥	刈	苅	瓦	乾	侃	冠	寒	刊	勘	勧	巻	喚	堪	姦
	3430	完	官	寛	干	幹	患	感	慣	憾	換	敢	柑	桓	棺	款	歓
	3440	汗	漢	澗	灌	環	甘	監	看	竿	管	簡	緩	缶	翰	肝	艦
	3450	莞	観	諫	貫	還	鑑	間	閑	関	陥	韓	館	舘	丸	含	岸
	3460	巌	玩	癌	眼	岩	翫	贋	雁	頑	顔	願					
キ	3460												企	伎	危	喜	器
	3470	基	奇	嬉	寄	岐	希	幾	忌	揮	机	旗	既	期	棋	棄	
	3520		機	帰	毅	気	汽	畿	祈	季	稀	紀	徽	規	記	貴	起
	3530	軌	輝	飢	騎	鬼	亀	偽	儀	妓	宜	戯	技	擬	欺	犠	疑
	3540	祇	義	蟻	誼	議	掬	菊	鞠	吉	吃	喫	桔	橘	詰	砧	杵
	3550	黍	却	客	脚	虐	逆	丘	久	仇	休	及	吸	宮	弓	急	救
	3560	朽	求	汲	泣	灸	球	究	窮	笈	級	糾	給	旧	牛	去	居
	3570	巨	拒	拠	挙	渠	虚	許	距	鋸	漁	禦	魚	亨	享	京	
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	3620		供	侠	僑	兇	競	共	凶	協	匡	卿	叫	喬	境	峡	強
	3630	彊	怯	恐	恭	挟	教	橋	況	狂	狭	矯	胸	脅	興	蕎	郷
	3640	鏡	響	饗	驚	仰	凝	尭	暁	業	局	曲	極	玉	桐	粁	僅
	3650	勤	均	巾	錦	斤	欣	欽	琴	禁	禽	筋	緊	芹	菌	衿	襟
	3660	謹	近	金	吟	銀											
ク	3660						九	倶	句	区	狗	玖	矩	苦	軀	駆	駈
	3670	駒	具	愚	虞	喰	空	偶	寓	遇	隅	串	櫛	釧	屑	屈	
	3720		掘	窟	沓	靴	轡	窪	熊	隈	粂	栗	繰	桑	鍬	勲	君
	3730	薫	訓	群	軍	郡											
ケ	3730						卦	袈	祁	係	傾	刑	兄	啓	圭	珪	型
	3740	契	形	径	恵	慶	慧	憩	掲	携	敬	景	桂	渓	畦	稽	系
	3750	経	継	繋	罫	茎	荊	蛍	計	詣	警	軽	頚	鶏	芸	迎	鯨
	3760	劇	戟	撃	激	隙	桁	傑	欠	決	潔	穴	結	血	訣	月	件
	3770	倹	倦	健	兼	券	剣	喧	圏	堅	嫌	建	憲	懸	拳	捲	
	3820		検	権	牽	犬	献	研	硯	絹	県	肩	見	謙	賢	軒	遣
	3830	鍵	険	顕	験	鹸	元	原	厳	幻	弦	減	源	玄	現	絃	舷
	3840	言	諺	限													
コ	3840				乎	個	古	呼	固	姑	孤	己	庫	弧	戸	故	枯
	3850	湖	狐	糊	袴	股	胡	菰	虎	誇	跨	鈷	雇	顧	鼓	五	互
	3860	伍	午	呉	吾	娯	後	御	悟	梧	檎	瑚	碁	語	誤	護	醐
	3870	乞	鯉	交	佼	侯	候	倖	光	公	功	効	勾	厚	口	向	
	3920		后	喉	坑	垢	好	孔	孝	宏	工	巧	巷	幸	広	庚	康
	3930	弘	恒	慌	抗	拘	控	攻	昂	晃	更	杭	校	梗	構	江	洪
	3940	浩	港	溝	甲	皇	硬	稿	糠	紅	紘	絞	綱	耕	考	肯	肱
	3950	腔	膏	航	荒	行	衡	講	貢	購	郊	酵	鉱	礦	鋼	閤	降
	3960	項	香	高	鴻	剛	劫	号	合	壕	拷	濠	豪	轟	麹	克	刻
	3970	告	国	穀	酷	鵠	黒	獄	漉	腰	甑	忽	惚	骨	狛	込	
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	3A20		此	頃	今	困	坤	墾	婚	恨	懇	昏	昆	根	梱	混	痕
	3A30	紺	艮	魂													
サ	3A30				些	佐	叉	唆	嵯	左	差	査	沙	瑳	砂	詐	鎖
	3A40	裟	坐	座	挫	債	催	再	最	哉	塞	妻	宰	彩	才	採	栽
	3A50	歳	済	災	采	犀	砕	砦	祭	斎	細	菜	裁	載	際	剤	在
	3A60	材	罪	財	冴	坂	阪	堺	榊	肴	咲	崎	埼	碕	鷺	作	削
	3A70	咋	搾	昨	朔	柵	窄	策	索	錯	桜	鮭	笹	匙	冊	刷	
	3B20		察	拶	撮	擦	札	殺	薩	雑	皐	鯖	捌	錆	鮫	皿	晒
	3B30	三	傘	参	山	惨	撒	散	桟	燦	珊	産	算	纂	蚕	讃	賛
	3B40	酸	餐	斬	暫	残											
シ	3B40						仕	仔	伺	使	刺	司	史	嗣	四	士	始
	3B50	姉	姿	子	屍	市	師	志	思	指	支	孜	斯	施	旨	枝	止
	3B60	死	氏	獅	祉	私	糸	紙	紫	肢	脂	至	視	詞	詩	試	誌
	3B70	諮	資	賜	雌	飼	歯	事	似	侍	児	字	寺	慈	持	時	
	3C20		次	滋	治	爾	璽	痔	磁	示	而	耳	自	蒔	辞	汐	鹿
	3C30	式	識	鴫	竺	軸	宍	雫	七	叱	執	失	嫉	室	悉	湿	漆
	3C40	疾	質	実	蔀	篠	偲	柴	芝	屡	蕊	縞	舎	写	射	捨	赦
	3C50	斜	煮	社	紗	者	謝	車	遮	蛇	邪	借	勺	尺	杓	灼	爵
	3C60	酌	釈	錫	若	寂	弱	惹	主	取	守	手	朱	殊	狩	珠	種
	3C70	腫	趣	酒	首	儒	受	呪	寿	授	樹	綬	需	囚	収	周	
	3D20		宗	就	州	修	愁	拾	洲	秀	秋	終	繍	習	臭	舟	蒐
	3D30	衆	襲	讐	蹴	輯	週	酋	酬	集	醜	什	住	充	十	従	戎
	3D40	柔	汁	渋	獣	縦	重	銃	叔	夙	宿	淑	祝	縮	粛	塾	熟
	3D50	出	術	述	俊	峻	春	瞬	竣	舜	駿	准	循	旬	楯	殉	淳
	3D60	準	潤	盾	純	巡	遵	醇	順	処	初	所	暑	曙	渚	庶	緒
	3D70	署	書	薯	藷	諸	助	叙	女	序	徐	恕	鋤	除	傷	償	
	3E20		勝	匠	升	召	哨	商	唱	嘗	奨	妾	娼	宵	将	小	少
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	3E30	尚	庄	床	廠	彰	承	抄	招	掌	捷	昇	昌	昭	晶	松	梢
	3E40	樟	樵	沼	消	渉	湘	焼	焦	照	症	省	硝	礁	祥	称	章
	3E50	笑	粧	紹	肖	菖	蒋	蕉	衝	裳	訟	証	詔	詳	象	賞	醤
	3E60	鉦	鍾	鐘	障	鞘	上	丈	丞	乗	冗	剰	城	場	壌	嬢	常
	3E70	情	擾	条	杖	浄	状	畳	穣	蒸	譲	醸	錠	嘱	埴	飾	
	3F20		拭	植	殖	燭	織	職	色	触	食	蝕	辱	尻	伸	信	侵
	3F30	唇	娠	寝	審	心	慎	振	新	晋	森	榛	浸	深	申	疹	真
	3F40	神	秦	紳	臣	芯	薪	親	診	身	辛	進	針	震	人	仁	刃
	3F50	塵	壬	尋	甚	尽	腎	訊	迅	陣	靭						
ス	3F50											笥	諏	須	酢	図	厨
	3F60	逗	吹	垂	帥	推	水	炊	睡	粋	翠	衰	遂	酔	錐	錘	随
	3F70	瑞	髄	崇	嵩	数	枢	趨	雛	据	杉	椙	菅	頗	雀	裾	
	4020		澄	摺	寸												
セ	4020					世	瀬	畝	是	凄	制	勢	姓	征	性	成	政
	4030	整	星	晴	棲	栖	正	清	牲	生	盛	精	聖	声	製	西	誠
	4040	誓	請	逝	醒	青	静	斉	税	脆	隻	席	惜	戚	斥	昔	析
	4050	石	積	籍	績	脊	責	赤	跡	蹟	碩	切	拙	接	摂	折	設
	4060	窃	節	説	雪	絶	舌	蝉	仙	先	千	占	宣	専	尖	川	戦
	4070	扇	撰	栓	栴	泉	浅	洗	染	潜	煎	煽	旋	穿	箭	線	
	4120		繊	羨	腺	舛	船	薦	詮	賎	践	選	遷	銭	銑	閃	鮮
	4130	前	善	漸	然	全	禅	繕	膳	糎							
ソ	4130										噌	塑	岨	措	曾	曽	楚
	4140	狙	疏	疎	礎	祖	租	粗	素	組	蘇	訴	阻	遡	鼠	僧	創
	4150	双	叢	倉	喪	壮	奏	爽	宋	層	匝	惣	想	捜	掃	挿	掻
	4160	操	早	曹	巣	槍	槽	漕	燥	争	痩	相	窓	糟	総	綜	聡
	4170	草	荘	葬	蒼	藻	装	走	送	遭	鎗	霜	騒	像	増	憎	
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	4220		臓	蔵	贈	造	促	側	則	即	息	捉	束	測	足	速	俗
	4230	属	賊	族	続	卒	袖	其	揃	存	孫	尊	損	村	遜		
タ	4230															他	多
	4240	太	汰	詑	唾	堕	妥	惰	打	柁	舵	楕	陀	駄	騨	体	堆
	4250	対	耐	岱	帯	待	怠	態	戴	替	泰	滞	胎	腿	苔	袋	貸
	4260	退	逮	隊	黛	鯛	代	台	大	第	醍	題	鷹	滝	瀧	卓	啄
	4270	宅	托	択	拓	沢	濯	琢	託	鐸	濁	諾	茸	凧	蛸	只	
	4320		叩	但	達	辰	奪	脱	巽	竪	辿	棚	谷	狸	鱈	樽	誰
	4330	丹	単	嘆	坦	担	探	旦	歎	淡	湛	炭	短	端	箪	綻	耽
	4340	胆	蛋	誕	鍛	団	壇	弾	断	暖	檀	段	男	談			
チ	4340														値	知	地
	4350	弛	恥	智	池	痴	稚	置	致	蜘	遅	馳	築	畜	竹	筑	蓄
	4360	逐	秩	窒	茶	嫡	着	中	仲	宙	忠	抽	昼	柱	注	虫	衷
	4370	註	酎	鋳	駐	樗	瀦	猪	苧	著	貯	丁	兆	凋	喋	寵	
	4420		帖	帳	庁	弔	張	彫	徴	懲	挑	暢	朝	潮	牒	町	眺
	4430	聴	脹	腸	蝶	調	諜	超	跳	銚	長	頂	鳥	勅	捗	直	朕
	4440	沈	珍	賃	鎮	陳											
ツ	4440						津	墜	椎	槌	追	鎚	痛	通	塚	栂	掴
	4450	槻	佃	漬	柘	辻	蔦	綴	鍔	椿	潰	坪	壷	嬬	紬	爪	吊
	4460	釣	鶴														
テ	4460			亭	低	停	偵	剃	貞	呈	堤	定	帝	底	庭	廷	弟
	4470	悌	抵	挺	提	梯	汀	碇	禎	程	締	艇	訂	諦	蹄	逓	
	4520		邸	鄭	釘	鼎	泥	摘	擢	敵	滴	的	笛	適	鏑	溺	哲
	4530	徹	撤	轍	迭	鉄	典	填	天	展	店	添	纏	甜	貼	転	顛
	4540	点	伝	殿	澱	田	電										
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ト	4540							兎	吐	堵	塗	妬	屠	徒	斗	杜	渡
	4550	登	菟	賭	途	都	鍍	砥	礪	努	度	土	奴	怒	倒	党	冬
	4560	凍	刀	唐	塔	塘	套	宕	島	嶋	悼	投	搭	東	桃	檮	棟
	4570	盗	淘	湯	濤	灯	燈	当	痘	禱	等	答	筒	糖	統	到	
	4620		董	蕩	藤	討	謄	豆	踏	逃	透	鐙	陶	頭	騰	闘	働
	4630	動	同	堂	導	憧	撞	洞	瞳	童	胴	萄	道	銅	峠	鴇	匿
	4640	得	徳	瀆	特	督	禿	篤	毒	独	読	栃	橡	凸	突	椴	届
	4650	鳶	苫	寅	酉	瀞	噸	屯	惇	敦	沌	豚	遁	頓	呑	曇	鈍
ナ	4660	奈	那	内	乍	凪	薙	謎	灘	捺	鍋	楢	馴	縄	畷	南	楠
	4670	軟	難	汝													
ニ	4670				二	尼	弐	邇	匂	賑	肉	虹	廿	日	乳	入	
	4720		如	尿	韮	任	妊	忍	認								
ヌ	4720									濡							
ネ	4720										禰	祢	寧	葱	猫	熱	年
	4730	念	捻	撚	燃	粘											
ノ	4730						乃	廼	之	埜	嚢	悩	濃	納	能	脳	膿
	4740	農	覗	蚤													
ハ	4740				巴	把	播	覇	杷	波	派	琶	破	婆	罵	芭	馬
	4750	俳	廃	拝	排	敗	杯	盃	牌	背	肺	輩	配	倍	培	媒	梅
	4760	楳	煤	狽	買	売	賠	陪	這	蝿	秤	矧	萩	伯	剥	博	拍
	4770	柏	泊	白	箔	粕	舶	薄	迫	曝	漠	爆	縛	莫	駁	麦	
	4820		函	箱	硲	箸	肇	筈	櫨	幡	肌	畑	畠	八	鉢	溌	発
	4830	醱	髪	伐	罰	抜	筏	閥	鳩	噺	塙	蛤	隼	伴	判	半	反
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ハ	4840	叛	帆	搬	斑	板	氾	汎	版	犯	班	畔	繁	般	藩	販	範
	4850	釆	煩	頒	飯	挽	晩	番	盤	磐	蕃	蛮					
ヒ	4850												匪	卑	否	妃	庇
	4860	彼	悲	扉	批	披	斐	比	泌	疲	皮	碑	秘	緋	罷	肥	被
	4870	誹	費	避	非	飛	樋	簸	備	尾	微	枇	毘	琵	眉	美	
	4920		鼻	柊	稗	匹	疋	髭	彦	膝	菱	肘	弼	必	畢	筆	逼
	4930	檜	姫	媛	紐	百	謬	俵	彪	標	氷	漂	瓢	票	表	評	豹
	4940	廟	描	病	秒	苗	錨	鋲	蒜	蛭	鰭	品	彬	斌	浜	瀕	貧
	4950	賓	頻	敏	瓶												
フ	4950					不	付	埠	夫	婦	富	冨	布	府	怖	扶	敷
	4960	斧	普	浮	父	符	腐	膚	芙	譜	負	賦	赴	阜	附	侮	撫
	4970	武	舞	葡	蕪	部	封	楓	風	葺	蕗	伏	副	復	幅	服	
	4A20		福	腹	複	覆	淵	弗	払	沸	仏	物	鮒	分	吻	噴	墳
	4A30	憤	扮	焚	奮	粉	糞	紛	雰	文	聞						
ヘ	4A30											丙	併	兵	塀	幣	平
	4A40	弊	柄	並	蔽	閉	陛	米	頁	僻	壁	癖	碧	別	瞥	蔑	箆
	4A50	偏	変	片	篇	編	辺	返	遍	便	勉	娩	弁	鞭			
ホ	4A50														保	舗	鋪
	4A60	圃	捕	歩	甫	補	輔	穂	募	墓	慕	戊	暮	母	簿	菩	倣
	4A70	俸	包	呆	報	奉	宝	峰	峯	崩	庖	抱	捧	放	方	朋	
	4B20		法	泡	烹	砲	縫	胞	芳	萌	蓬	蜂	褒	訪	豊	邦	鋒
	4B30	飽	鳳	鵬	乏	亡	傍	剖	坊	妨	帽	忘	忙	房	暴	望	某
	4B40	棒	冒	紡	肪	膨	謀	貌	貿	鉾	防	吠	頬	北	僕	卜	墨
	4B50	撲	朴	牧	睦	穆	釦	勃	没	殆	堀	幌	奔	本	翻	凡	盆
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
マ	4B60	摩	磨	魔	麻	埋	妹	昧	枚	毎	哩	槙	幕	膜	枕	鮪	柾
	4B70	鱒	桝	亦	俣	又	抹	末	沫	迄	侭	繭	麿	万	慢	満	
	4C20		漫	蔓													
ミ	4C20				味	未	魅	巳	箕	岬	密	蜜	湊	蓑	稔	脈	妙
	4C30	粍	民	眠													
ム	4C30				務	夢	無	牟	矛	霧	鵡	椋	婿	娘			
メ	4C30														冥	名	命
	4C40	明	盟	迷	銘	鳴	姪	牝	滅	免	棉	綿	緬	面	麺		
モ	4C40															摸	模
	4C50	茂	妄	孟	毛	猛	盲	網	耗	蒙	儲	木	黙	目	杢	勿	餅
	4C60	尤	戻	籾	貰	問	悶	紋	門	匁							
ヤ	4C60										也	冶	夜	爺	耶	野	弥
	4C70	矢	厄	役	約	薬	訳	躍	靖	柳	薮	鑓					
ユ	4C70												愉	愈	油	癒	
	4D20		諭	輸	唯	佑	優	勇	友	宥	幽	悠	憂	揖	有	柚	湧
	4D30	涌	猶	猷	由	祐	裕	誘	遊	邑	郵	雄	融	夕			
ヨ	4D30														予	余	与
	4D40	誉	輿	預	傭	幼	妖	容	庸	揚	揺	擁	曜	楊	様	洋	溶
	4D50	熔	用	窯	羊	耀	葉	蓉	要	謡	踊	遥	陽	養	慾	抑	欲
	4D60	沃	浴	翌	翼	淀											
ラ	4D60						羅	螺	裸	来	莱	頼	雷	洛	絡	落	酪
	4D70	乱	卵	嵐	欄	濫	藍	蘭	覧								
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
リ	4D70									利	吏	履	李	梨	理	璃	
	4E20		痢	裏	裡	里	離	陸	律	率	立	葎	掠	略	劉	流	溜
	4E30	琉	留	硫	粒	隆	竜	龍	侶	慮	旅	虜	了	亮	僚	両	凌
	4E40	寮	料	梁	涼	猟	療	瞭	稜	糧	良	諒	遼	量	陵	領	力
	4E50	緑	倫	厘	林	淋	燐	琳	臨	輪	隣	鱗	麟				
ル	4E50													瑠	塁	涙	累
	4E60	類															
レ	4E60		令	伶	例	冷	励	嶺	怜	玲	礼	苓	鈴	隷	零	霊	麗
	4E70	齢	暦	歴	列	劣	烈	裂	廉	恋	憐	漣	煉	簾	練	聯	
	4F20		蓮	連	錬												
ロ	4F20					呂	魯	櫓	炉	賂	路	露	労	婁	廊	弄	朗
	4F30	楼	榔	浪	漏	牢	狼	籠	老	聾	蝋	郎	六	麓	禄	肋	録
	4F40	論															
ワ	4F40		倭	和	話	歪	賄	脇	惑	枠	鷲	亙	亘	鰐	詫	藁	蕨
	4F50	椀	湾	碗	腕												
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

J．予約語

ここにあげた予約語は，**N80-BASIC，N-BASIC** で，特定の意味を持ち，変数名などに使うことはできません。

A
AND
ABS
ATN
ASC
AUTO
ATTR$

B
BCD$
BEEP

C
CONSOLE
CLOSE
CONT
CLEAR
CLOAD
CSAVE
CSRLIN
CINT
CSNG
CDBL
CVI
CVS
CVD
COS
CHR$
CMD
COLOR

D
DATA
DIM
DEFSTR
DEFINT
DEFSNG
DEFDBL
DSKO$
DEF
DELETE
DSKI$
DSKF
DEC
DATE$

E
END
ELSE
ERASE
ERROR
ERL
ERR
EXP
EOF
EQV

F
FORMAT
FOR
FIELD
FILES
FN
FRE
FIX
FPOS

G
GOTO
GO TO
GOSUB
GET

H
HEX$

I
INPUT
IF
INSTR
INT
INP
IMP
INIT
INKEY$
ISET
IRESET
IEEE

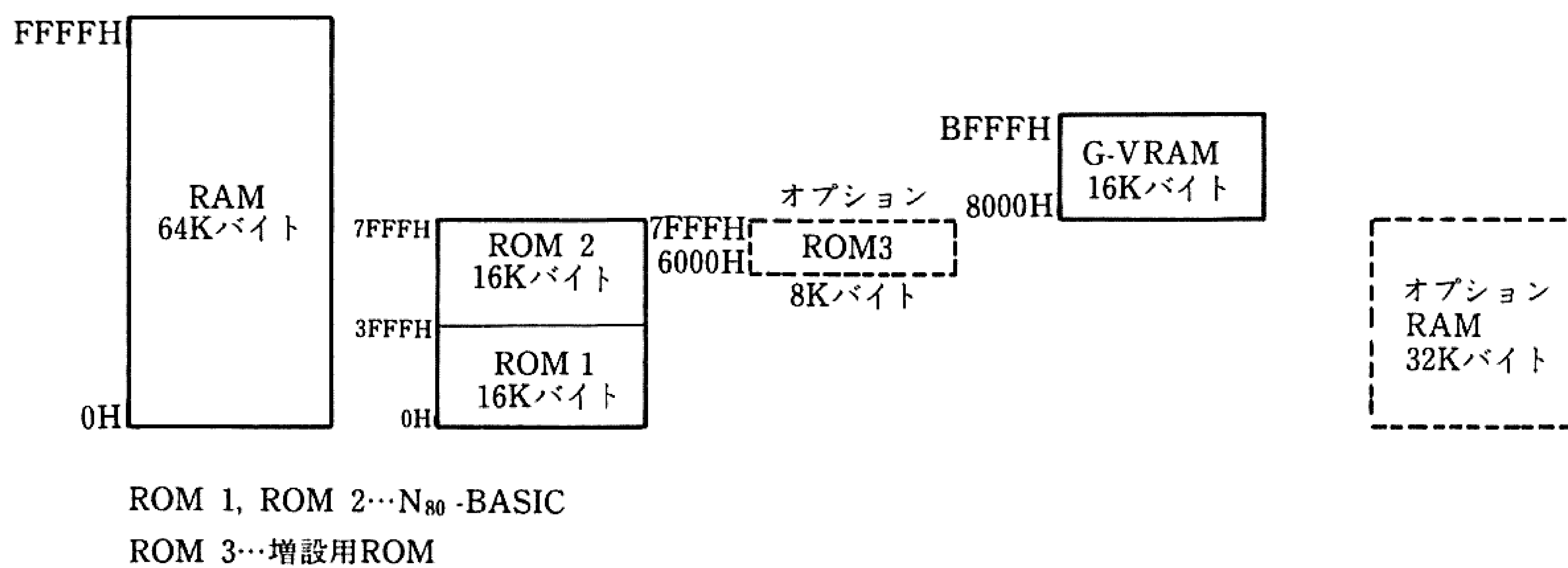
K
KILL
KEY

L
LET
LOCATE
LINE
LOAD
LSET
LPRINT
LLIST
LPOS
LISTEN
LIST
LFILES
LOG
LOC
LEN
LEFT$
LOF

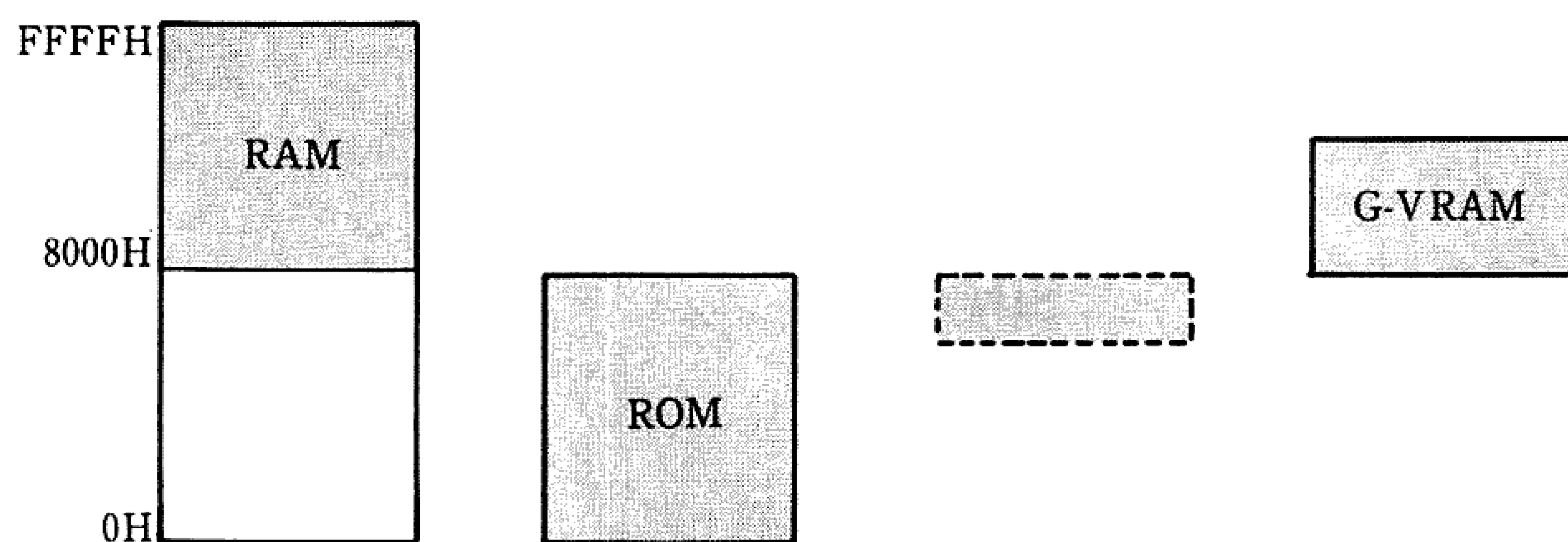
M
MOUNT
MERGE
MOD

	MKI$		RUN		TAB(
	MKS$		RESTONE		TO
	MKD$		RETURN		THEN
	MID$		REMOVE		TAN
	MOTOR		REM		TERM
	MON		RESUME		TALK
	MAT		RSET		TIME$
N	NEXT		RIGHT$	U	USING
	NAME		RND		USR
	NEW		RENUM	V	VAL
	NOT		RBYTE		VARPTR
O	OUT	S	STOP	W	WIDTH
	ON		SWAP		WAIT
	OPEN		SET		WBYTE
	OR		SAVE	X	XOR
	OCT$		SPC(		+
P	PUT		STEP		−
	POKE		SGN		*
	PRINT		SQR		/
	POS		SIN		^
	PEEK		STR$		¥
	PORT		STRING$		"
	POLL		SPACE$		>
	PSET		STATUS		=
	PRESET		SRQ		<
	POINT	T	TRON		
R	READ		TROFF		

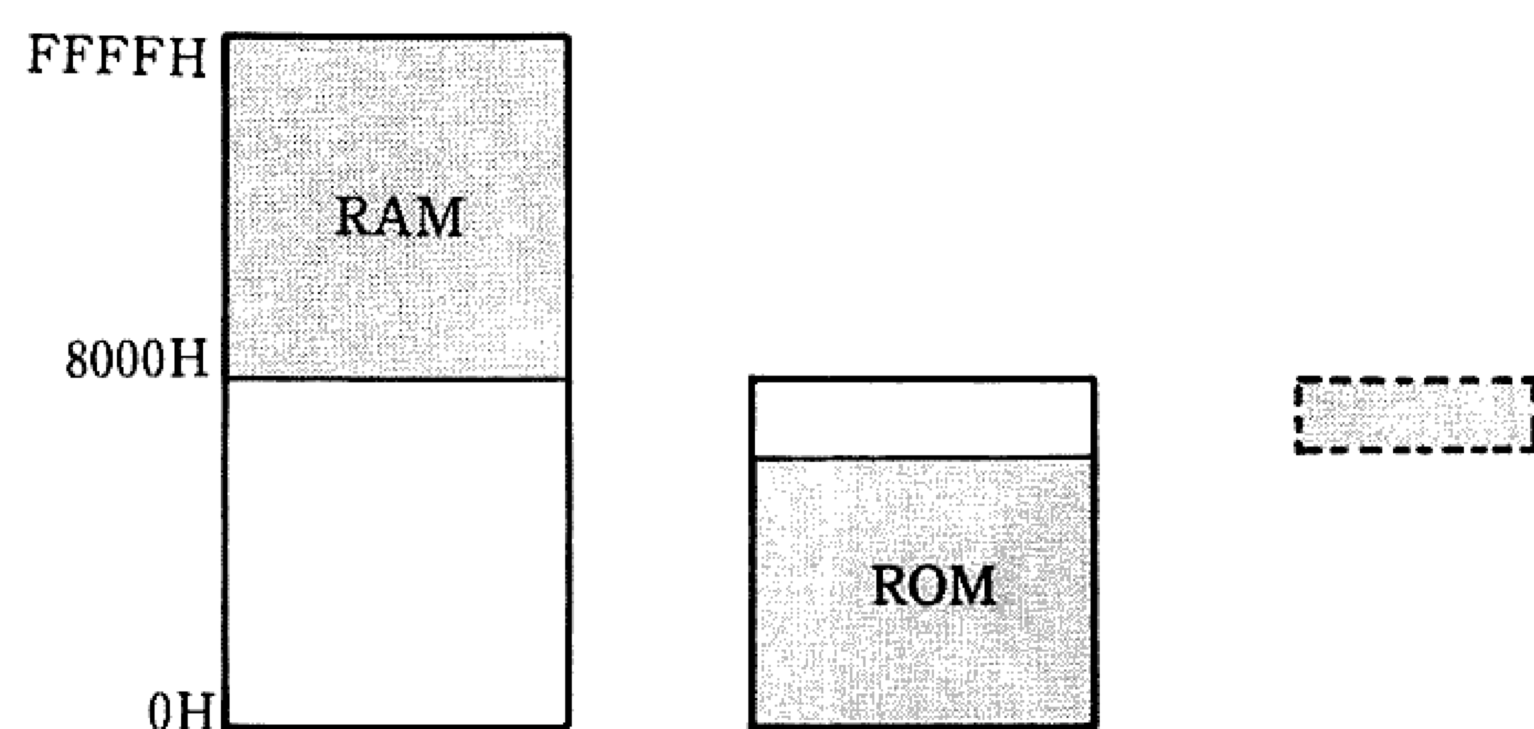
K. メモリマップ



ROM 1, ROM 2…N_{80}-BASIC
ROM 3…増設用ROM



N_{80}-BASICで使われるメモリ



N-BASICで使われるメモリ

L.　μPD780ニーモニック↔機械語対照表

8ビットロード

X	I	R	A	B	C	D	E	H	L	(HL)	(BC)	(DE)	(IX+d)	(IY+d)	(nn)	n
LD A, X	ED 57	ED 5F	7F	78	79	7A	7B	7C	7D	7E	0A	1A	DD 7E d	FD 7E d	3A n n	3E n
LD B, X			47	40	41	42	43	44	45	46			DD 46 d	FD 46 d		06 n
LD C, X			4F	48	49	4A	4B	4C	4D	4E			DD 4E d	FD 4E d		0E n
LD D, X			57	50	51	52	53	54	55	56			DD 56 d	FD 56 d		16 n
LD E, X			5F	58	59	5A	5B	5C	5D	5E			DD 5E d	FD 5E d		1E n
LD H, X			67	60	61	62	63	64	65	66			DD 66 d	FD 66 d		26 n
LD L, X			6F	68	69	6A	6B	6C	6D	6E			DD 6E d	FD 6E d		2E n
LD (HL), X			77	70	71	72	73	74	75							36 n
LD (BC), X			02													
LD (DE), X			12													
LD (IX+d), X			DD 77 d	DD 70 d	DD 71 d	DD 72 d	DD 73 d	DD 74 d	DD 75 d							DD 36 d n
LD (IY+d), X			FD 77 d	FD 70 d	FD 71 d	FD 72 d	FD 73 d	FD 74 d	FD 75 d							FD 36 d n
LD (nn), X			32 n n													
LD I, X			ED 47													
LD R, X			ED 4F													

16ビットロード

X	AF	BC	DE	HL	SP	IX	IY	nn	(nn)
LD AF, X									
LD BC, X								01 n n	ED 4B n n
LD DE, X								11 n n	ED 5B n n
LD HL, X								21 n n	2A n n
LD SP, X				F9		DD F9	FD F9	31 n n	ED 7B n n
LD IX, X								DD 21 n n	DD 2A n n
LD IY, X								FD 21 n n	FD 2A n n
LD (nn), X		ED 43 n n	ED 53 n n	22 n n	ED 73 n n	DD 22 n n	FD 22 n n		
PUSH X	F5	C5	D5	E5		DD E5	FD E5		
POP X	F1	C1	D1	E1		DD E1	FD E1		

ブロック転送

LDI	ED A0
LDIR	ED B0
LDD	ED A8
LDDR	ED B8

ブロック・サーチ

CPI	ED A1
DPIR	ED B1
CPD	ED A9
CPDR	ED B9

8ビット算術論理演算

X	A	B	C	D	E	H	L	(HL)	(IX+d)	(IY+d)	n
ADD A, X	87	80	81	82	83	84	85	86	DD 86 d	FD 86 d	C6 n
ADC A, X	8F	88	89	8A	8B	8C	8D	8E	DD 8E d	FD 8E d	CE n
SUB X	97	90	91	92	93	94	95	96	DD 96 d	FD 96 d	D6 n
SBC A, X	9F	98	99	9A	9B	9C	9D	9E	DD 9E d	FD 9E d	DE n
AND X	A7	A0	A1	A2	A3	A4	A5	A6	DD A6 d	FD A6 d	E6 n
XOR X	AF	A8	A9	AA	AB	AC	AD	AE	DD AE d	FD AE d	EE n
OR X	B7	B0	B1	B2	B3	B4	B5	B6	DD B6 d	FD B6 d	F6 n
CP X	BF	B8	B9	BA	BB	BC	BD	BE	DD BE d	FD BE d	FE n
INC X	3C	04	0C	14	1C	24	2C	34	DD 34 d	FD 34 d	
DEC X	3D	05	0D	15	1D	25	2D	35	DD 35 d	FD 35 d	

CPUコントロール

NOP	00
HALT	76
DI	F3
EI	FB
IM 0	ED 46
IM 1	ED 56
IM 2	ED 5E

16ビット算術演算

X	BC	DE	HL	SP	IX	IY
ADD HL, X	09	19	29	39		
ADD IX, X	DD 09	DD 19		DD 39	DD 29	
ADD IY, X	FD 09	FD 19		FD 39		FD 29
ADC HL, X	ED 4A	ED 5A	ED 6A	ED 7A		
SBC HL, X	ED 42	ED 52	ED 62	ED 72		
INC X	03	13	23	33	DD 23	FD 23
DEC X	0B	1B	2B	3B	DD 2B	FD 2B

エクスチェンジ

EX AF, AF′	08
EX DE, HL	EB
EX (SP), HL	E3
EX (SP), IX	DD E3
EX (SP), IY	FD E3
EXX	D9

アキュムレータ操作

DAA	27
CPL	2F
NEG	ED 44
CCF	3F
SCF	37

ローテート，シフト

X	A	B	C	D	E	H	L	(HL)	(IX +d)	(IY +d)
RLC X	CB 07	CB 00	CB 01	CB 02	CB 03	CB 04	CB 05	CB 06	DD CB d 06	FD CB d 06
RRC X	CB 0F	CB 08	CB 09	CB 0A	CB 0B	CB 0C	CB 0D	CB 0E	DD CB d 0E	FD CB d 0E
RL X	CB 17	CB 10	CB 11	CB 12	CB 13	CB 14	CB 15	CB 16	DD CB d 16	FD CB d 16
RR X	CB 1F	CB 18	CB 19	CB 1A	CB 1B	CB 1C	CB 1D	CB 1E	DD CB d 1E	FD CB d 1E
SLA X	CB 27	CB 20	CB 21	CB 22	CB 23	CB 24	CB 25	CB 26	DD CB d 26	FD CB d 26
SRA X	CB 2F	CB 28	CB 29	CB 2A	CB 2B	CB 2C	CB 2D	CB 2E	DD CB d 2E	FD CB d 2E
SRL X	CB 3F	CB 38	CB 39	CB 3A	CB 38	CB 3C	CB 3D	CB 3E	DD CB d 3E	FD CB d 3E
RLD								ED 6F		
RRD								ED 67		

	A
RLCA	07
RRCA	0F
RLA	17
RRA	1F

ジャンプ，コール，リターン

X	UN COND	C	NC	Z	NZ	PE	PO	M	P	
JP X, nn	C3 n n	DA n n	D2 n n	CA n n	C2 n n	EA n n	E2 n n	FA n n	F2 n n	
JR X, e	18 e−2	38 e−2	30 e−2	28 e−2	20 e−2					
JP (HL)	E9									
JP (IX)	DD E9									
JP (IY)	FD E9									
CALL X, nn	CD n n	DC n n	D4 n n	CC n n	C4 n n	EC n n	E4 n n	FC n n	F4 n n	
DJNZ e										10 e−2
RET X	C9	D8	C0	C8	C0	E8	E0	F8	F0	
RETI	ED 4D									
RETN	ED 45									

リスタート

RST 00H	C7
RST 08H	CF
RST 10H	D7
RST 18H	DF
RST 20H	E7
RST 28H	EF
RST 30H	F7
RST 38H	FF

ビット操作

X	A	B	C	D	E	H	L	(HL)	(IX+d)	(IY+d)
BIT 0, X	CB 47	CB 40	CB 41	CB 42	CB 43	CB 44	CB 45	CB 46	DD CB d 46	FD CB d 46
BIT 1, X	CB 4F	CB 48	CB 49	CB 4A	CB 4B	CB 4C	CB 4D	CB 4E	DD CB d 4E	FD CB d 4E
BIT 2, X	CB 57	CB 50	CB 51	CB 52	CB 53	CB 54	CB 55	CB 56	DD CB d 56	FD CB d 56
BIT 3, X	CB 5F	CB 58	CB 59	CB 5A	CB 5B	CB 5C	CB 5D	CB 5E	DD CB d 5E	FD CB d 5E
BIT 4, X	CB 67	CB 60	CB 61	CB 62	CB 63	CB 64	CB 65	CB 66	DD CB d 66	FD CB d 66
BIT 5, X	CB 6F	CB 68	CB 69	CB 6A	CB 6B	CB 6C	CB 6D	CB 6E	DD CB d 6E	FD CB d 6E
BIT 6, X	CB 77	CB 70	CB 71	CB 72	CB 73	CB 74	CB 75	CB 76	DD CB d 76	FD CB d 76
BIT 7, X	CB 7F	CB 78	CB 79	CB 7A	CB 7B	CB 7C	CB 7D	CB 7E	DD CB d 7E	FD CB d 7E
RES 0, X	CB 87	CB 80	CB 81	CB 82	CB 83	CB 84	CB 85	CB 86	DD CB d 86	FD CB d 86
RES 1, X	CB 8F	CB 88	CB 89	CB 8A	CB 8B	CB 8C	CB 8D	CB 8E	DD CB d 8E	FD CB d 8E
RES 2, X	CB 97	CB 90	CB 91	CB 92	CB 93	CB 94	CB 95	CB 96	DD CB d 96	FD CB d 96
RES 3, X	CB 9F	CB 98	CB 99	CB 9A	CB 9B	CB 9C	CB 9D	CB 9E	DD CB d 9E	FD CB d 9E

X	A	B	C	D	E	H	L	(HL)	(IX +d)	(IY +d)
RES 4, X	CB A7	CB A0	CB A1	CB A2	CB A3	CB A4	CB A5	CB A6	DD CB d A6	FD CB d A6
RES 5, X	CB AF	CB A8	CB A9	CB AA	CB AB	CB AC	CB AD	CB AE	DD CA d AE	FD CB d AE
RES 6, X	CB B7	CB B0	CB B1	CB B2	CB B3	CB B4	CB B5	CB B6	DD CB d B6	FD CB d B6
RES 7, X	CB BF	CB B8	CB B9	CB BA	CB BB	CB BC	CB BD	CB BE	DD CB d BE	FD CB d BE
SET 0, X	CB C7	CB C0	CB C1	CB C2	CB C3	CB C4	CB C5	CB C6	DD CB d C6	FD CB d C6
SET 1, X	CB CF	CB C8	CB C9	CB CA	CB CB	CB CC	CB CD	CB CE	DD CB d CE	FD CB d CE
SET 2, X	CB D7	CB D0	CB D1	CB D2	DB D3	CB D4	DB D5	CB D6	DD CB d D6	FD CB d D6
SET 3, X	CB FD	CB D8	CB D9	CB DA	CB DB	CB DC	CB DD	CB DE	DD CB d DE	FD CB d DE
SET 4, X	CB E7	CB E0	CB E1	CB E2	CB E3	CB E4	CB E5	CB E6	DD CB d E6	FD CB d E6
SET 5, X	CB EF	CB E8	CB E9	CB EA	CB EB	CB EC	CB ED	CB EE	DD CB d EE	FD CB d EE
SET 6, X	CB F7	CB F0	CB F1	CB F2	CB F3	CB F4	CB F5	CB F6	DD CB d F6	FD CB d F6
SET 7, X	CB FF	CB F8	CB F9	CB FA	CB FB	CB FC	CB FD	CB FE	DD CB d FE	FD CB d FE

入力

IN A, n	DB n
IN A, (C)	ED 78
IN B, (C)	ED 40
IN C, (C)	ED 48
IN D, (C)	ED 50
IN E, (C)	ED 58
IN H, (C)	ED 60
IN L, (C)	ED A2
INI	ED A2
INR	ED B2
IND	ED AA
INDR	ED BA

出力

OUT n, A	D3 n
OUT (C), A	ED 79
OUT (C), B	ED 41
OUT (C), C	ED 49
OUT (C), D	ED 51
OUT (C), E	ED 59
OUT (C), H	ED 61
OUT (C), L	ED 69
OUTI	ED A3
OTIR	ED B3
OUTD	ED AB
OTDR	ED BB

M．機械語のサブルーチン

ROM に書き込まれている機械語のサブルーチンのいくつかをあげます。

機　　　能	サブルーチンの使い方
画面に１文字表示する。	Aレジスタにキャラクタコードを入れてから CALL 257Hを実行する。
キー入力があるまで待って，１文字入力されたら戻る。	CALL F75Hを実行すると，Aレジスタに入力されたキーのキャラクタコードを入れて戻る。
キー入力があるかどうかを調べる。	CALL F7BHを実行すると，キー入力があればAレジスタに入力されたキーのキャラクタコード，キー入力がなければキャリフラグをセットして戻る。
プリンタに１文字出力する。	Aレジスタにキャラクタコードを入れてから CALL D60Hを実行する。
文字列を画面に表示する。	HLレジスタに文字列の先頭アドレスを入れ，CALL 52EDHを実行する。文字列の最後には，00Hを付ける。
BASICのコマンド入力待ち状態にする。（ウォームスタート）	JP 8Hを実行する。
モニタのコマンド入力待ち状態にする。	JP 5C66Hを実行する。
画面に〝？〟を表示してモニタのコマンド入力待ち状態にする。	JP 5C2CHを実行する。

これらのサブルーチンの使い方は，PC-8001MKIIユーザーズマニュアルの第10章モニタのＳコマンドとＧコマンドの使い方のところで示したサンプルプログラムを参照して下さい。

N. 高解像度グラフィックスとモード

CMD SCREEN の起動時の設定値

CMD SCREEN　　0,0,7

CMD COLOR の起動時の設定値

CMD COLOR　　3,0

グラフィックモード / カラーナンバ	モノクロモード	アトリビュートカラーモード	4色カラーモード0	4色カラーモード1
0	黒	選択色(黒)	黒	青
1	選択色(白)	テキスト画面に依存します。	赤	マゼンタ
2	———	———	緑	シアン
3	———	———	選択色(青)	選択色(黒)

O. カラーコードに対応する色(カラーモード)と機能(白黒モード)

(CONSOLE, COLOR参照)

カラーコード	色(カラーモード)	機能(白黒モード)
0	黒	ノーマル
1	青	シークレット
2	赤	ブリンク
3	マゼンタ	シークレット
4	緑	リバース
5	シアン	リバースシークレット
6	黄	リバースブリンク
7	白	リバースシークレット

P. 誘導関数

N_{80}(N)-BASIC が“組み込み”関数として用意していない三角関数のうち，いくつかは，用意された関数を使って作り出すことができます。以下の数式を参考にしてください(誤差の範囲に注意が必要です)。

目的とする関数	組み込み関数からの誘導式
セカント	SEC(X)= 1/COS(X)
コセカント	COSEC(X)=1/SIN(X)
コタンジェント	COT(X)=1/TAN(X)
アークサイン	ARCSIN(X)=ATN(X/SQR(−X*X+1))
アークコサイン	ARCCOS(X)=−ATN(X/SQR(−X*X+1)) +1.5708
アークセカント	ARCSEC(X)=ATN(SQR(X*X−1))+(SGN (X)−1)*1.5708
アークコセカント	ARCCOSEC(X)=ATN(1/SQR(X*X−1)) +(SGN(X)−1*1.5708
アークコタンジェント	ARCCOT(X)=−ATN(X)+1.5708
ハイパーボリック・サイン	SINH(X)=(EXP(X)−EXP(−X))/2
ハイパーボリック・コサイン	COSH(X)= EXP(−X)/2)
ハイパーボリック・タンジェント	TANH(X)=−EXP(−X)/(EXP(X)+EXP (−X))*2+1
ハイパーボリック・セカント	SECH(X)=2/(EXP(X)+EXP(−X))
ハイパーボリック・コセカント	COSECH(X)=2/(EXP(X)−EXP(−X))
ハイパーボリック・コタンジェント	COTH(X)=EXP(−X)/(EXP(X)−EXP (−X))*2+1
ハイパーボリック・アークサイン	ARCSINH(X)=LOG(X+SQR(X*X+1))
ハイパーボリック・アークコサイン	ARCCOSH(X)=LOG(X+SQR(X*X−1))
ハイパーボリック・アークタンジェント	ARCTANH(X)=LOG((1 +X)/(1−X))/2
ハイパーボリック・アークセカント	ARCSECH(X)=LOG((SQR(−X * X+1)+1) /X)
ハイパーボリック・アークコセカント	ARCCOSECH(X)=LOG((SGN(X)*SQR(X* X+1)+1)/X)
ハイパーボリック・アークコタンジェント	ARCCOTH(X)=LOG((X+1)/(X−1))/2

Q．ソフトウェア上の一般的注意事項

1. PC-8001用のプログラムを使用するときは，ディップスイッチ8をonにセットしてN-BASICモードで使用してください。

   PC-8001用のプログラムをN_{80}-BASICモードで使用した場合，正常に動作しないことがあります。その原因として考えられることは，①機械語のサブルーチンなどが，N_{80}BASICのワークエリアを使用している，②プログラムのサイズや，変数領域が大きすぎて(大きな配列を使用している等)，実行するとOut of memoryエラーを生ずる，などがあります。

2. PC-8001用のプログラムのうち割り込みを使っているものはMKIIをN-BASICモードにしても使えないことがあります。

3. N_{80}-BASIC, N_{80}DISK-BASICにおいて，6000H番地から7FFFH番地に書かれた機械語プログラムをUSR関数で呼びその値をCMDのついた命令とSTATUSのついた関数の中で使うような場合は，①のようにします。②のように直接使うことはできません。

   ① 正しい例

   ```
   10 DEFUSR=&H6808
   15 TMP=USR(0)
   20 CMD VIEW(TMP,1)-(100,100)
   ```

   ② 誤っている例

   ```
   10 DEFUSR=&H6808
   20 CMD VIEW(USR(0),1)-(100,100)
   ```

4. 市販のPC-8001用のプログラムで，ブザーを使っているものの中に，カレンダ時計を狂わせてしまうものがあります。このようなときは，もう一度カレンダ時計の値を設定すれば正しい状態に直ります。

5. **N_{80}DISK-BASIC**では，ファイルをオープンしたときには，フロッピィディスクを取り出す前に必ず**CLOSE**してください。**CLOSE**せずにフロッピィディスクを取り出すとファイルがこわれてしまい

ます。

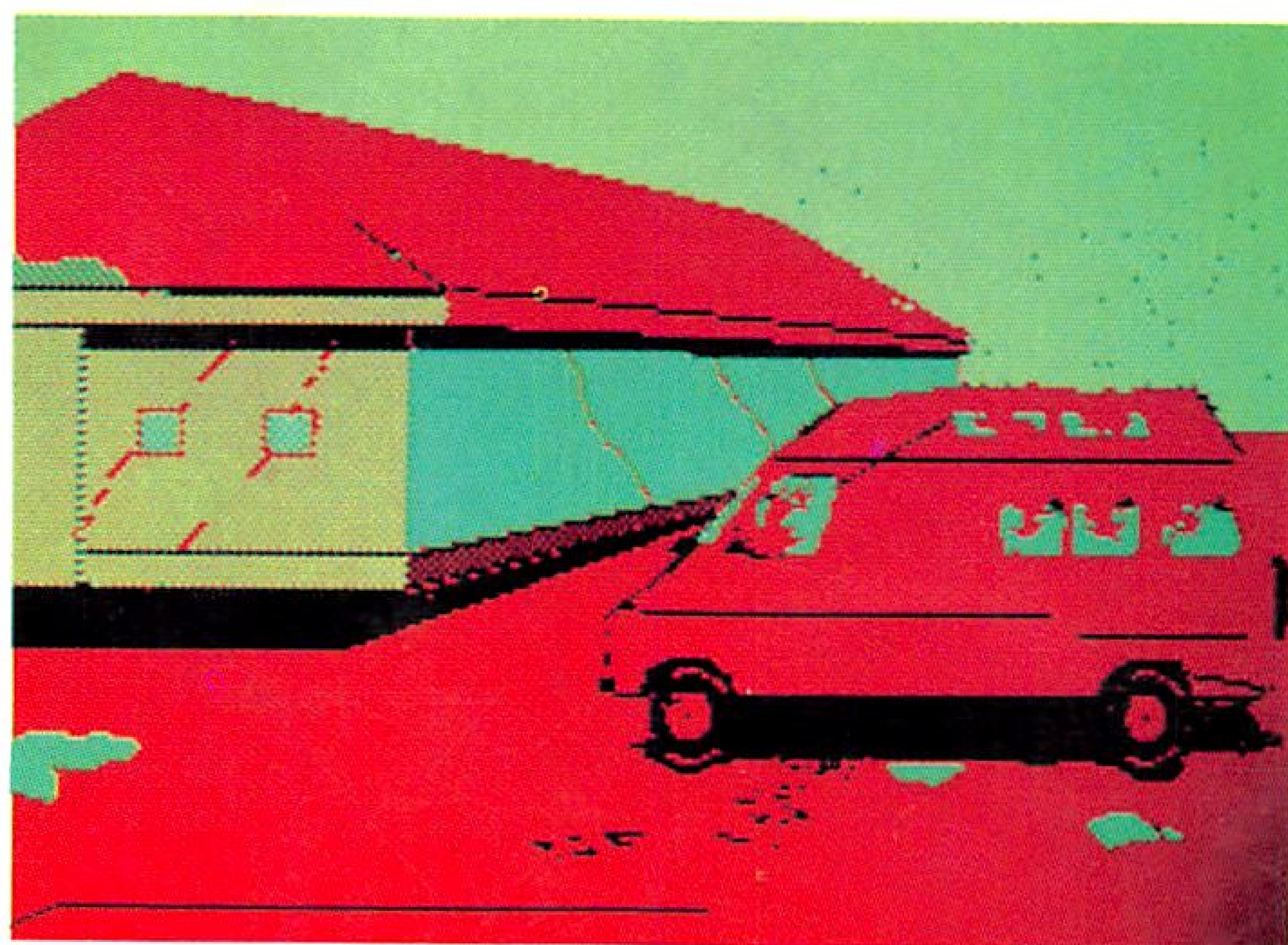
6. **DISK-BASIC** ではフロッピィディスクを取り出す前に必ず **REMOVE** を実行してください。**REMOVE** せずにフロッピィディスクを取り出すとファイルがこわれてしまいます。

7. カセットテープにプログラムをセーブしたときは，必ずベリファイしてください。ベリファイは，セーブした直後に行います。セーブした後，RUN してしまうとその後ベリファイしてもメモリ上のプログラムとカセットテープ上のプログラムは一致しないことがあります。

8. カセットテープにセーブするプログラムの最後には必ず end をつけてください。end がないと，そのプログラムをロードしても実行できないことがあります。

9. ターミナルモードを使用するときボーレートを 600 bps 以上に設定すると，受信エラーになることがあります。

10. N_{80}DISK-BASIC でプログラムの実行中またはロード中に “Out of memory” エラーが出るときは，起動時に “How many files ?” に対して答える「同時にオープンするファイル数」を減らしてください。



NEC 日本電気株式会社・新日本電気株式会社

78113